USER'S GUIDE

Descartes® Route Planner™ Background Optimizer (BGO)

May 2017

Descartes® Route Planner™ Background Optimizer

The Descartes Systems Group Inc.

120 Randall Drive

Waterloo, Ontario Canada, N2V 1C6

Phone: 519-746-8110

Internet: http://www.descartes.com

Customer Support

In North America: 1-877-786-9339

Outside North America: +800 -7866-3390

e-mail: support@descartes.com

Table of Contents

Document Conventions

This document uses the following conventions:

- Names of windows, frames, dialogs, menus, list boxes, and lists begin with uppercase and are bolded. (Tools menu, Save button)
- Key combinations that you press appear in mixed case. If the keys are joined by a plus sign (+), press and hold the first key simultaneously with the remaining keys (for example, CTRL+ALT+DEL).
- Text that you type appears in `Courier New` font. (Enter `USERID` in the login field.)
- Cross-references to other documents, or to sections within the current document, appear in underlined italics. (See *Saving a File* for details.)
- *Italics* are used for emphasis throughout this document.

➲ **Note—** Information important to a particular task or function is introduced with the note format and icon.

ⓘ **Tip—** Information that may make completing a task easier, but isn't essential to the task, is introduced with the tip format and icon.

⚠ **Warning— This warning format indicates information that you need to pay particular attention to. Ignoring information presented as a warning could lead to damage and unexpected results. Disregarding information presented as a warning may result in damage to your software or data.**

Introduction

What is the BGO

The Background Optimizer (BGO) is a separate component of the Descartes® Route Planner™ application. BGO is a routing and scheduling optimization engine that reviews and optimizes routes on a recurring basis to try and improve the routes. It can be setup to use different settings at different times of the day.

A company typically has several BGOs setup to work on the routes in the Descartes Route Planner database. One instance of a BGO is comprised of two windows services (one Scheduler service and one Optimizer service). The routes in the database to be optimized are grouped into Data Slices and a BGO optimizes one data slice at a time. For the BGO to work efficiently the data slices should select no more than 2000-2500 orders and they must all be in the same Schedule Key. A typical implementation will have several BGOs with many data slices.

BGO Optimization Methodology

The BGO optimizes the data selected by a Data Slice independently (see *What is a Data Slice* under *Installing and Configuring BGO* for more information). If any improvements are found in the routes then they are submitted back to Descartes Route Planner immediately for review and if accepted the change is applied to the route(s) in the database. Descartes Route Planner will reject the change if there isn't an increase in the profit of the route(s) or if the route(s) have been changed by another process while the BGO was optimizing those routes. So changes made through the UI or via API calls to Descartes Route Planner take precedence over improvements proposed by the BGO. Each optimization performed by the BGO needs to be quick otherwise there is a greater chance of it being rejected because the route was changed by another process.

The BGO is made up of two parts: the Scheduler and the Optimizer. The Scheduler is a service that holds the Data Slice information (routes and stops) in memory and performs optimizations as directed by the Optimizer service. The Optimizer service inspects the routes in Scheduler and determines what optimizations should be run by scheduler to try and improve the route quality.

There are two main types of optimization: Batch and Incremental. Batch optimization is when you pass the optimizer engine (RMPI) a big group of routes and orders and tell it to optimize them. This can take RMPI quite a while to do. Incremental optimization is when you just pass RMPI a few routes at a time or one unassigned job at a time to work on. The BGO is usually setup to run incremental optimizations but it can be setup to perform batch optimizations.

The BGO has four different processes that it can be configured to perform.

- Optimizing a Data Slice using small, quick optimizations.

- Optimize all the routes in a Data Slice using an "Optimize All" command to Scheduler.
- Send a Refresh Routes command XML to Descartes Route Planner
- Send Command XML to Descartes Route Planner.

Optimizing a Data Slice.

The Optimizer has seven different small incremental optimization phases that it can run for predetermined periods of time. These phases optimize the routes in slightly different ways, but all phases should allow for route quality improvements. The BGO continues to cycle round performing these optimizations until it is time to move on to another data slice.

The seven optimization phases are:

- Intra-route phase. This phase resequences routes one at a time. It inspects the ordering of jobs on the route and attempts to improve the sequencing of them.
- Batch Assignment / Inter phase. This phase attempts to move orders between a select number of compatible routes to improve profitability. For Batch Assign, a group of compatible routes is selected, then the orders on those routes are unassigned and the optimizer engine called to assign them back onto the same set of routes. The BGO can be set to include all the unassigned orders as well or just those that were originally on the group of routes. If no improvement in profitability is found the BGO can then reinstate the routes as they were and run just an Inter optimization on them. If Batch Assign optimizations are turned off and just Inter is turned on then the group of compatible routes is formed and just an Inter optimization run on it.

  Compatible routes are:

  - Based on date_shift property (Set in the Optimizer settings file) which determines how far apart the EarliestStartDate of the routes can be to be included in a group
  - Based on Dynamic Zoning distance
  - Based on Compatible Territories (territory swaps are used if set).

  Note that Requirements are not a criteria when forming groups of compatible routes
- Assignment phase. This phase attempts to assign any unassigned orders to routes. This will take care of any orders created and not assigned to routes for any reason. The BGO can assign unassigned orders one at a time or in clusters. It can also assign in descending profit order if profit is being used to determine the priority of an order.
- Check Multiple Routes Servicing Same Location.

The BGO checks if more than one route on a day is going to the same latitude and longitude. It then, internally, moves all the orders for that location to each route and selects the one that increases the profit the most. (Or decreases it the least). That improvement is sent to Descartes Route Planner. Now this may well cause a violation because other BGO optimizations should have moved the orders to the same route if profitable. So Descartes Route Planner must be configured to allow a decrease in profitability. (CtySysValues.AllowableProfitDrop).

- Check Order Date Violations

  If this phase is turned on then the BGO will check stops with a window violation and if the order is on a route for the wrong date it is unassigned. So if one stop of a double ended or complex order is being serviced on the wrong date then the order is unassigned. The Schedule UseTimeWindows setting is taken into account and the windows on the location checked if necessary.

- Fill Routes
  This phase takes an existing route and finds unassigned orders close to the stops already on the route. This Fill phase is meant for implementations where there are many more unassigned orders than can be serviced each day. Routes are created/seeded with orders that must be serviced (high priority) and then the route is filled with the best/closest lower priority orders. A number of related settings ensure that existing stops furthest away from the depot are used first to find unassigned orders close to them. Unassigned orders close to the depot are easy to assign since there are more resources that can service them.

- Assign Matching.

  New optimization phase to assign unassigned orders by finding another order for the same location key that is already assigned to a route and moving the unassigned order before or after it on that route. The order is moved to the route using a command xml call to Route Planner to move it. (DocRouteEditTask). So this can generate violations.

  Current restrictions:-

  1. Only single ended orders are assigned.
  2. The date of the Begun time of the existing order must match one of the windows of the unassigned order. Times are not checked.
  3. The route must have any requirements on the unassigned order.
  4. If the unassigned order is a dropoff it is placed before the existing order. If it is a pickup it is placed after.
  5. The match is on location key not latitude and longitude. (Possible future enhancement).

Performing an Optimize All

The BGO can be set to run an Optimize All on all the routes and stops in the data slice. This optimization can take a long time so the BGO should only ever be set to do this if other processes are not changing the routes at this time. Otherwise the result of the Optimize All could be rejected by Descartes Route Planner. This Optimize All is run in the scheduler service and so does not lock the database unlike one run from the UI. The BGO optimize all can optionally, internally, unassign all the orders first.

Send Refresh Routes API to Descartes Route Planner

The BGO can be set to send an API call to Descartes Route Planner to refresh all the routes selected by the data slice. When this setting is enabled, action will be performed once, when the data slice is first loaded by the BGO instance. By default, this setting is disabled.

When using external routing, the BGO will build the cache before sending the refresh command XML. If no other BGO optimization phases are turned on, the BGO will move on to the next data slice after refreshing. This feature is useful if map edits have been changed to create new cache files and a refresh is needed for the time and distance of existing routes in the database.

Sending Command xml and Business Documents.

The BGO can be configured to post a file of xml to any Descartes Route Planner listener. The typical uses for this are to generate resources, move routes from one schedule to another or run an optimization in Descartes Route Planner rather than in the BGO's Scheduler service. These xml commands have no relation to the data slice that the BGO is set to optimize. Typically the BGO is set to send the xml at a certain time of day and after that it goes back to optimizing a data slice continuously until it is time to send the command xml again.

Posting/Exporting Optimization Statistics and Route Information

A BGO instance optimizes a data slice for a certain length of time and then moves on to another one. At this point it can be configured to output information about the optimizations it has just performed and the routes and orders in the data slice. This information can be posted to a URL and/or written to a file.

How Descartes Route Planner and BGO Work Together

Descartes Route Planner BGO work together to improve the efficiency of the routes held in the Descartes Route Planner database. Sets of routes and orders, called Data Slices, are optimized by the BGO. A Data Slice can only select routes and orders from one schedule key, but there can be many data slices. If a schedule key contains a large number of routes and orders then a Data Slice might only select some of them. It might select on date or territory or another criteria.

Installing and Configuring BGO

What is a Data Slice

Data slices are defined in the following Descartes Route Planner database tables:

- FWDataSetGroup
- FWDataSetNesting
- FWDataSetValue

They can be configured through the Descartes Route Planner UI by selecting **Setup > Dataset >** (**Data Slice**, **Nestings**, or **Values**) from the main menu. These tables are used by Descartes Route Planner to create a SQL statement to select Routes and Stops for a particular Schedule Key to pass to the BGO. The routes can be selected on any attribute of the FWRoute or FWResource table so a typical selection might be by ScheduleKey, Territory, and EarliestStartDate.

The selection might be "select all routes where ScheduleKey='DefaultSchedule' and Territory='NY' and EarliestStartDate<=today+4". When a route is selected all the stops/orders on that route are automatically selected too.

Configuring a Data Slice

Data Set Groups

There is one row in the FWDataSetGroup table for each data slice. The only attributes that need to be setup for each row are:

- the DataSetGroupKey, which can be any unique descriptive name.
- the Type, which is either BGO or DEDICATED_BGO.
    - **Dedicated BGO:** A dedicated BGO has a one-to-one relationship with a data slice that has also been configured as dedicated. The dedicated BGO will only optimize the segment of the schedule as defined by that data slice and will continue to do so until the BGO is stopped.
    - **Dynamic BGO (Non-dedicated):** A typical implementation has many Data Slices and it is not practical, nor necessary, to have the same number of BGOs as Data Slices. A standard BGO is setup to request a Data Slice from Descartes Route Planner and then work on the routes and orders selected for a configurable length of time. At the end of that time, it will request a new Data Slice from Descartes Route Planner. Descartes Route Planner responds to these requests by cycling round the Data Slices and passing the Data Slice that is not currently being optimized and that was optimized the longest ago. The number of Data Slices and BGOs are decided at implementation time.

- The area name where the default is –NoAreas-. Multiple Data Slices of type BGO can be given the same area name and one or more BGOs can be configured to optimize only Data Slices with a particular name.

The following are optional fields.

- **Active:** When this setting is enabled, the BGO will optimize the data slice. This functionality applies to both types of data slices: dedicated and dynamic.

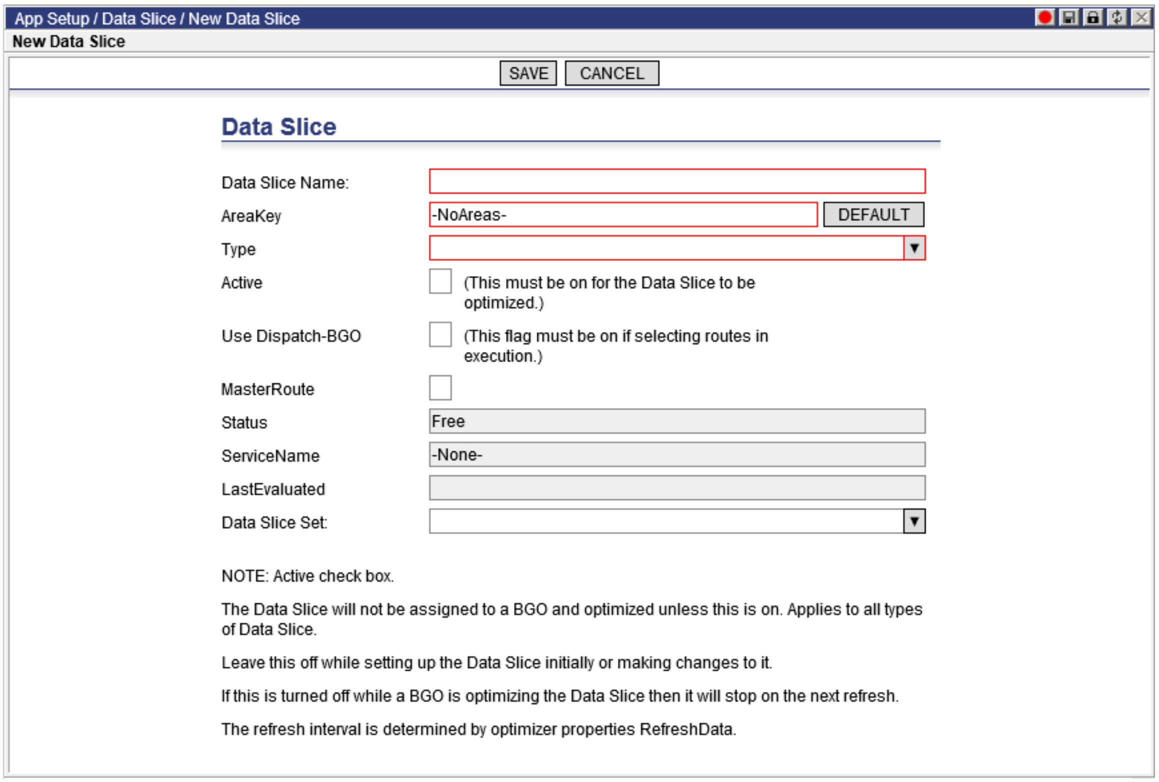
  The **Active** setting is disabled by default in order to allow users to configure the data slice first and then activate it when configuration is complete. In cases where the data slice is active and being optimized and this setting is turned off, the system will behave as follows:

  - The data slice will not be assigned to a BGO instance.
  - Or, if a BGO instance is currently optimizing the data slice, the next time that the BGO requests a refresh of the data slice, the data slice will not be returned, causing BGO instance to move on to another data slice. The refresh interval is controlled by the RefreshData optimizer property:

    com.descartes.escheduler.optimizer.refresh_lnosfw_data

  ➲ **Note—** Already existing slices will be automatically marked as active during the upgrade process.

- **@UseDispatchBGO:** Specifies whether or not this data slice selects from a Dispatching schedule
- **Data Slice Set:** Links the data slice to optimizer properties. (See the section on BGO Properties in the database.



Data Slice

Data Slice Name:	
AreaKey	-NoAreas-
Type	
Active	(This must be on for the Data Slice to be optimized.)
Use Dispatch-BGO	(This flag must be on if selecting routes in execution.)
MasterRoute	
Status	Free
ServiceName	-None-
LastEvaluated	
Data Slice Set:	

NOTE: Active check box.

The Data Slice will not be assigned to a BGO and optimized unless this is on. Applies to all types of Data Slice.

Leave this off while setting up the Data Slice initially or making changes to it.

If this is turned off while a BGO is optimizing the Data Slice then it will stop on the next refresh.

The refresh interval is determined by optimizer properties RefreshData.

Data Set Nesting

The rows in the FWDataSetNesting table define the attribute names to be used in the selection. Each data slice refers to at least three nesting rows; one to select the schedule settings, another to select routes, and a third to select unassigned orders. All data slices will use the same nesting row for selecting the schedule. The Descartes Route Planner install comes with 3 FWDataSetNesting records already setup that will select all routes and unassigned in a schedulekey.

- DefaultBGOSchedule selects the schedule settings.
- DefaultBGORoute selects the routes.
- DefaultBGOStop selects the unassigned orders.

These can be used or new records generated that select on more than just the schedule key. If all the Data Slices are going to select routes and stops on the same set of attributes, then only one nesting row needs to be generated to select routes and another for unassigned stops. The data slices will all link to these rows.

Example. To select routes on schedule key, territory and EarliestStartDate then the following FWDataSetNesting record must be generated.

Route selection:

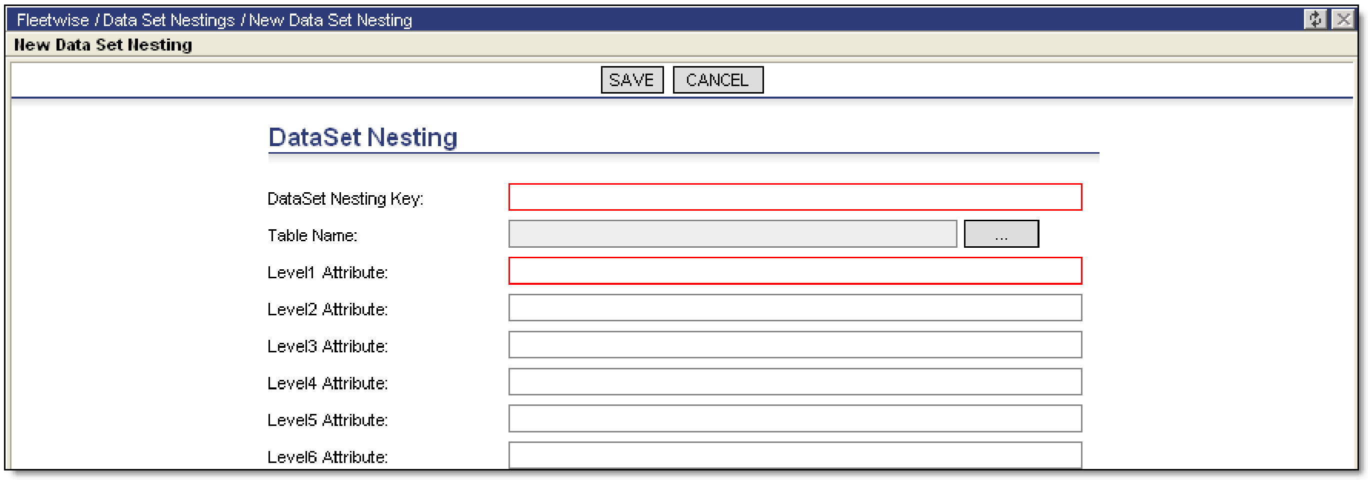
Set the DataSetNestingKey to a unique descriptive value.

Set Table Name to FWResource
Set Level1 Attribute to ScheduleKey
Set Level2 Attribute to Territory
Set Level3 Attribute to EarliestStartDate

To select unassigned orders on schedulekey and territory this FWDataSetNesting record must be generated.

Unassigned Orders:

Set the DataSetNestingKey to a unique descriptive value.
Set the Table Name to FWStop
Set the Level1 Attribute to ScheduleKey
Set the Level2 Attribute to Territory



DataSet Nesting

DataSet Nesting Key:	
Table Name:	
Level1 Attribute:	
Level2 Attribute:	
Level3 Attribute:	
Level4 Attribute:	
Level5 Attribute:	
Level6 Attribute:	

Data Set Values

FWDataSetValue table rows contain the values that match the attribute names defined in the nesting rows. A SQL SELECT statement is built by combining value and nesting rows. The value row contains the value of an attribute and the operator to be used with it in the SQL SELECT statement. Each data slice will have at least three rows in this table; one for selecting the schedule, another for the routes, and the third for the unassigned stops. They can have multiple rows selecting routes and unassigned stops.

String and date time values must have single quotes around them. Operators can be any valid SQL operator such as >, >=, =, IN, etc. If the operator IN is used, then the value must have a parenthesis around it.

There are three special operators for selecting dates relative to today: DATE>=, DATE<=, and DATE=. For the special operators the value should be the number of days from today.

There are two special operators for selecting fields with a null value; INWITHNULL and =WITHNULL. INWITHNULL is basically the SQL IN operator but it will also select the record if the field is null. Similarly =WITHNULL selects the field if it is null or = to the given value.

➲ **Note—** The special operators are case sensitive.

To setup the FWDataSetValues to select the schedule settings:-

For the Schedule:

Set the DataSetGroupKey using the Browse button to select the FWDataSetGroup table.
Set the DataSetNestingsKey using the Browse button to select from the FWDataSetNesting table.
Set the End level to the number of attributes being used. For the schedule, just one.
Set the Level1Value to the name of the schedule key enclosed in single quotes.
Set the Level1Operator to the operator to be used. For the schedule, it will always be =.

To setup the FWDataSetValues to select routes on schedule key, territory and date

For the Routes:

Set the DataSetGroupKey using the Browse button to select the FWDataSetGroup table.
Set the DataSetNestingKey using the Browse button to select from the FWDataSetNesting table.
Set the End level to three since we are using three attributes in this example.
Set the Level1Value to the name of the Schedule Key enclosed in single quotes.
Set the Level2Value to the name of the territory enclosed in single quotes.
Set the Level3Value to the number of days relative to today. In this example, four (4).
Set the Level1Operator to =.
Set the Level2Operator to =.
Set the Level3Operator to DATE<=.

For the Stops:

Set the DataSetGroupKey using the Browse button to select the FWDataSetGroup table.
Set the DataSetNestingKey using the Browse button to select the FWDataSetNesting table.
Set the End level to two since we are using two attributes in this example.
Set the Level1Value to the name of the Schedule Key enclosed in quotes.
Set the Level2Value to the name of the territory enclosed in single quotes.
Set the Level1Operator to =.
Set the Level2Operator to =.

Examples of combined nesting and values rows.

Attribute	Operator	Value	Resulting SQL query
ScheduleKey	=	'DefaultSchedule'	ScheduleKey = 'DefaultSchedule'
Territory	IN	('Territory1', 'Territory2', …)	Territory IN ('Territory1', 'Territory2', …)
Measure1	>	10000	Measure1 > 10000
EarliestStartDate	DATE>=	3	DateDiff(day, DateAdd(day, 3, getdate()), EarliestStartDate) >= 0

All these queries will 'AND' together in the SELECT statement.

- A data slice can have multiple FWDataSetValue rows linked to the same FWDataSetNesting row. An 'OR' Relationship will be used between them in the SQL SELECT statement.

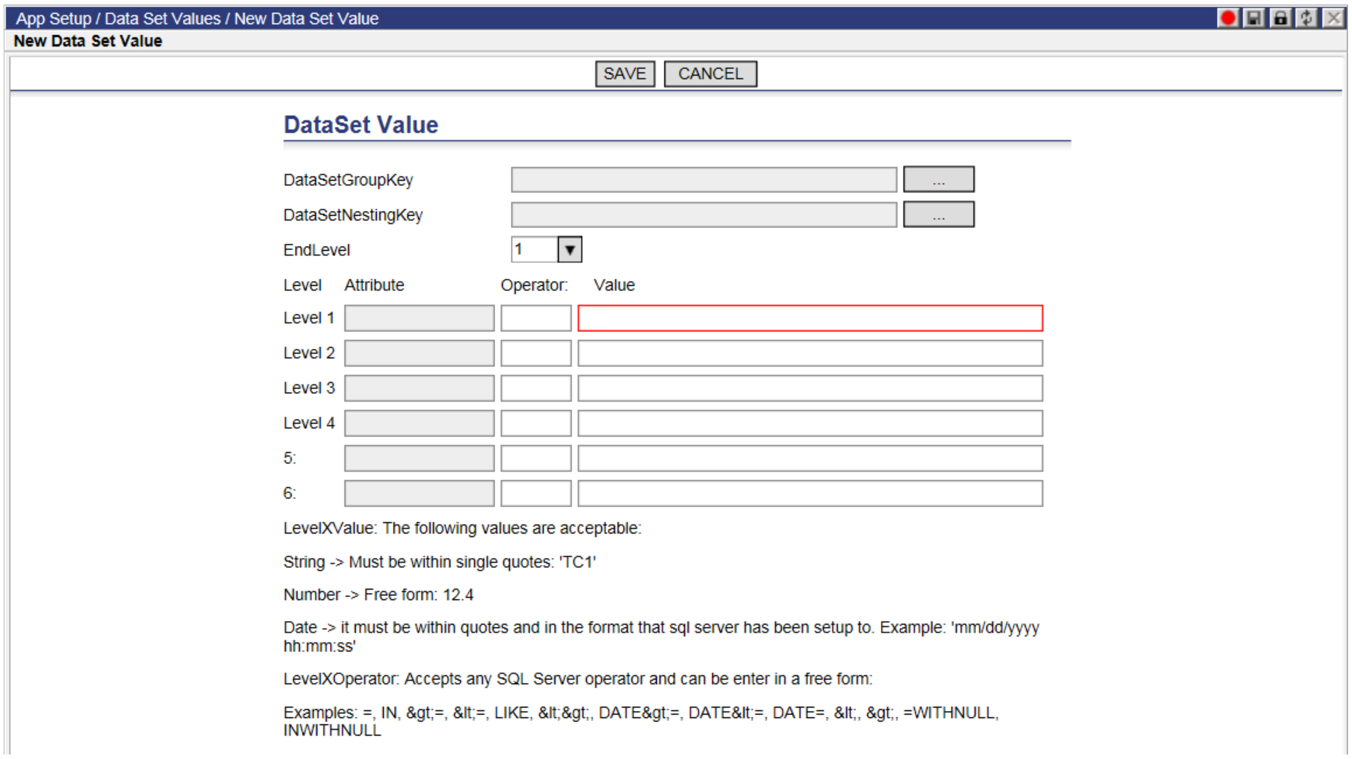
If there is a row in the FWDataSetNesting table like this:

DataSetGroupKey	Level1Attribute	Level2Attribute
"Zone_Req"	"Territory"	"Requirements"

And there are two rows in the FWDataSetValue table like this:

Group Key	Nesting Key	Level1 Value	Level1 Operator	Level2 Value	Level2 Operator
DataSlice1	Zone_Req	'Zone1'	=	'GAS'	=
DataSlice1	Zone_Req	'Zone2'	=	'GAS'	=

The resulting where clause will be like this: (Territory = 'Zone1' AND Requirements='GAS') OR (Territory = 'Zone2' AND Requirements='GAS')



DataSetGroupKey			
DataSetNestingKey			
EndLevel	1		
Level	Attribute	Operator:	Value
Level 1			
Level 2			
Level 3			
Level 4			
5:			
6:			

LevelXValue: The following values are acceptable:

String -> Must be within single quotes: 'TC1'

Number -> Free form: 12.4

Date -> it must be within quotes and in the format that sql server has been setup to. Example: 'mm/dd/yyyy hh:mm:ss'

LevelXOperator: Accepts any SQL Server operator and can be enter in a free form:

Examples: =, IN, &gt;=, &lt;=, LIKE, &lt;&gt;, DATE&gt;=, DATE&lt;=, DATE=, &lt;, &gt;, =WITHNULL, INWITHNULL

Introduction to BGO Instances

What are BGO Instances

A BGO instance is defined as an installation of the java scheduler and the BGO client.

If the computer has dual or quad processors, then multiple instances of the BGO can be installed on it. One instance of the BGO requires one processor to run efficiently. In the Descartes Route Planner installer, you can specify more than one "area" name. A second instance of the BGO generates another area folder under Fleetwise/scheduler/conf and two more windows services. The windows service names have the name of the area folder in them so that you know which go with each area folder.

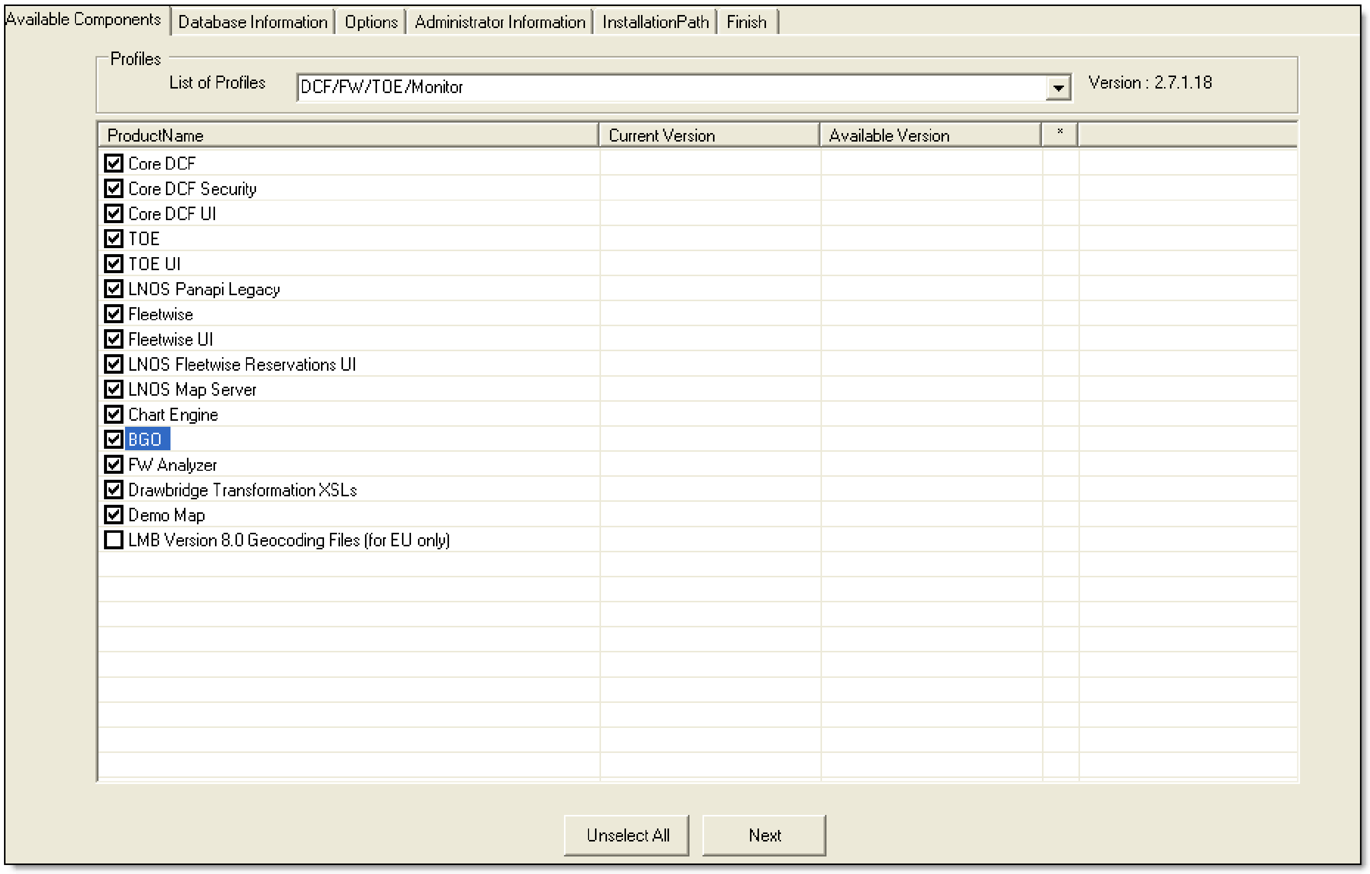
Installing BGO Components

Installation of BGO

For new installation of the BGO or to update an existing installation or add more instances, follow these instructions.

1 Start the Descartes Route Planner Installer.

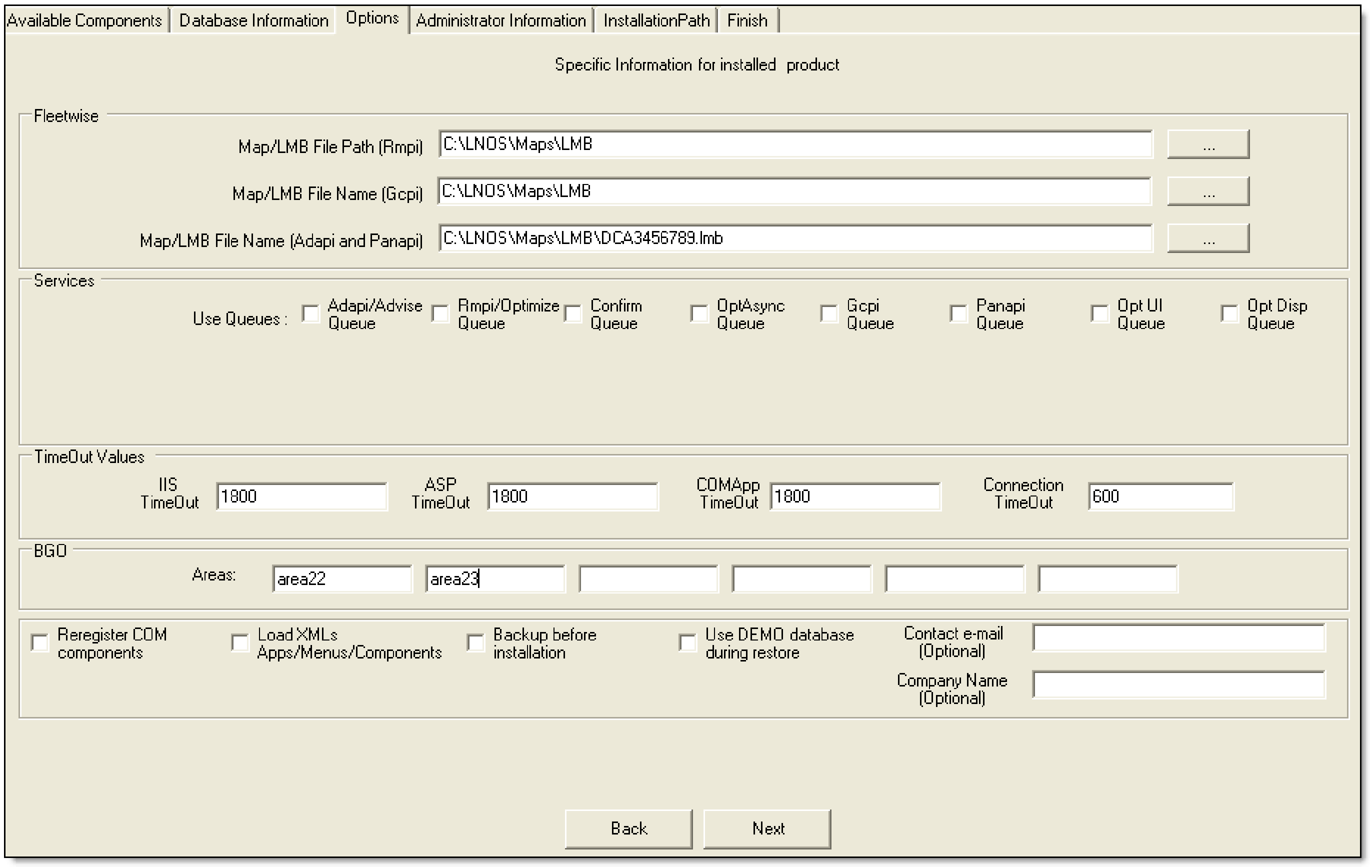
2 On the **Available Components** tab, make sure that the BGO check box is selected.



List of Profiles: DCF/FW/TOE/Monitor
Version : 2.7.1.18

ProductName	Current Version	Available Version	*
✓ Core DCF			
✓ Core DCF Security			
✓ Core DCF UI			
✓ TOE			
✓ TOE UI			
✓ LNOS Panapi Legacy			
✓ Fleetwise			
✓ Fleetwise UI			
✓ LNOS Fleetwise Reservations UI			
✓ LNOS Map Server			
✓ Chart Engine			
✓ BGO			
✓ FW Analyzer			
✓ Drawbridge Transformation XSLs			
✓ Demo Map			
☐ LMB Version 8.0 Geocoding Files (for EU only)			

3 On the **Options** tab, choose any area name you want for the BGO area fields.



Specific Information for installed product

Fleetwise

Map/LMB File Path (Rmpi): C:\LNOS\Maps\LMB

Map/LMB File Name (Gcpi): C:\LNOS\Maps\LMB

Map/LMB File Name (Adapi and Panapi): C:\LNOS\Maps\LMB\DCA3456789.lmb

Services

Use Queues : ☐ Adapi/Advise Queue ☐ Rmpi/Optimize Queue ☐ Confirm Queue ☐ OptAsync Queue ☐ Gcpi Queue ☐ Panapi Queue ☐ Opt UI Queue ☐ Opt Disp Queue

TimeOut Values

IIS TimeOut: 1800 | ASP TimeOut: 1800 | COMApp TimeOut: 1800 | Connection TimeOut: 600

BGO

Areas: area22 | area23

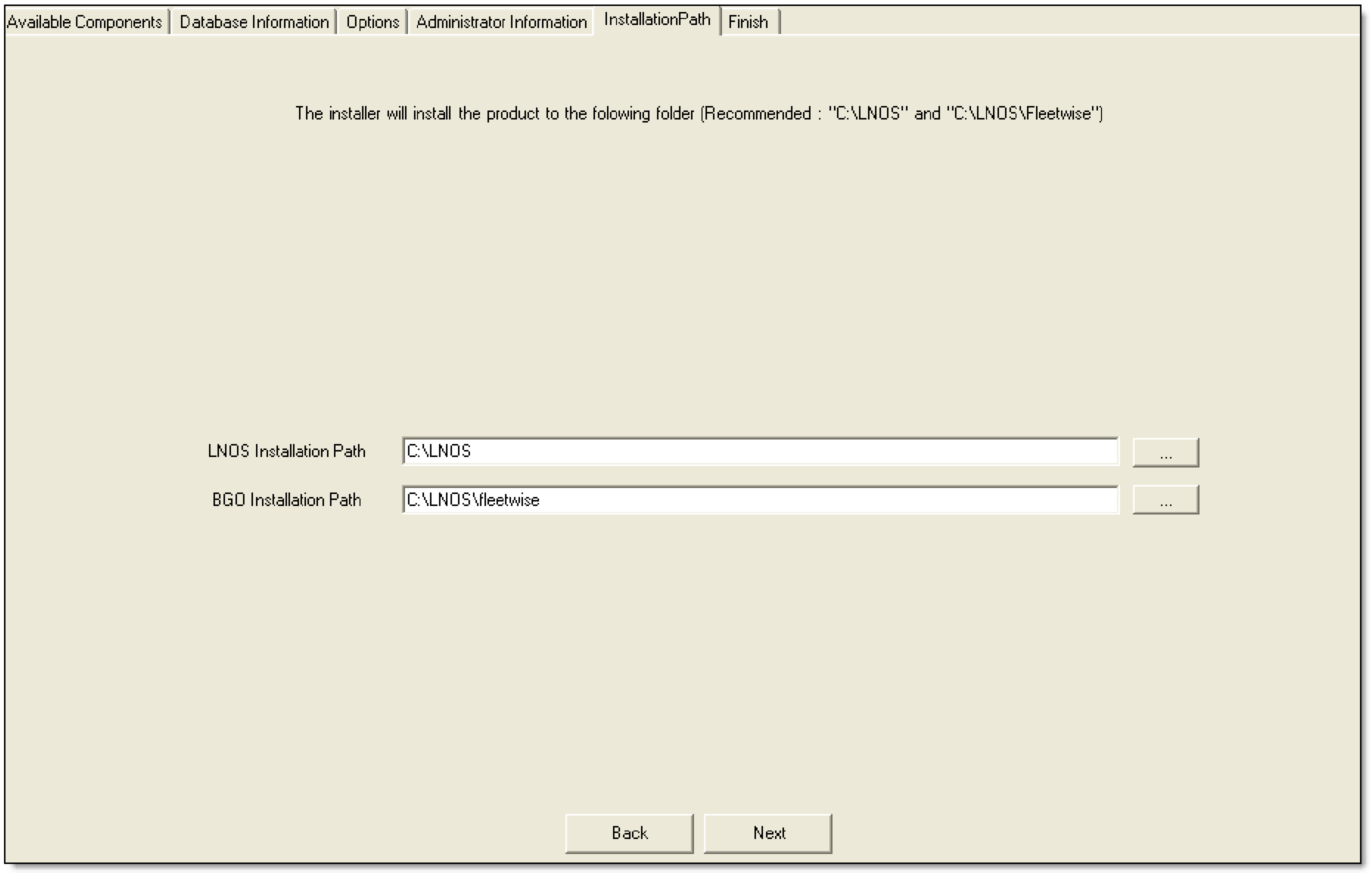
☐ Reregister COM components ☐ Load XMLs Apps/Menus/Components ☐ Backup before installation ☐ Use DEMO database during restore

Contact e-mail (Optional):

Company Name (Optional):

➲ **Note—** In the screenshot above, 'area22' and 'area23' are just examples. However, you do not have to have area in the name

4 On the **InstallationPath** tab, enter the path to the root folder for the scheduler install in the BGO Installation Path field. The default is c:\lnos\Fleetwise.



The installer will install the product to the folowing folder (Recommended : "C:\LNOS" and "C:\LNOS\Fleetwise")

LNOS Installation Path: C:\LNOS

BGO Installation Path: C:\LNOS\fleetwise

5 On the **Finish** tab, click **Finish**.

Explanation of Installed Components

When BGO is installed, a java Scheduler is installed. One BGO comprises a java Scheduler and a BGO client that is using that scheduler. The Descartes Route Planner installer lets you specify where the folder hierarchy is installed and the base folder name. Typically the folder is called Fleetwise. After installation, you end up with various sub folders under the Fleetwise folder, one of which is called scheduler. A scheduler windows service is registered and that generates an "area" folder under Fleetwise/scheduler/conf. The area folder(s) are named what you specified on the Options tab during Descartes Route Planner installation. In the "area" folder are the properties files that control the two services: local.properties and optimizer.properties.

BGO Configuration Files and Properties

Configuration of the BGO is made using the files:

- Local.properties
  - Scheduler Service Settings
- Optimizer.properties
  - Optimizer Service Settings

These files can be found in the area folder for each BGO.

The "optimizer.properties" control:

- The different sets of properties to use at different times of the day. They also control when to sleep and not perform any optimizations.
- Which of the four optimization phases should be run and the length of time that each individual optimization is allowed to run before it is halted.
- The length of time the optimizer spends running Intra, assigning unassigned orders and Batch Assign/Inter optimizations before moving on to the next kind of optimization.
- Which Resources/Routes to be optimized based on a Resource attribute and the zone. (This is not normally used.)
- Which unassigned jobs should be assigned, the jobs can be selected by a Job attribute and by zone. (This is not normally used)
- During the Intra, Batch Assign, and Inter phases, the routes that have changed can be optimized first and then the rest.

The "Local.properties" control:

- The url of Descartes Route Planner to use.
- The BGO type, DEDICATED or not.
- The map locations.
- The details for logging in to Descartes Route Planner, Company, name and password.

Local.properperites

When the Scheduler reads the local.properties file on startup, there are two kinds of properties in the file:

- those whose key starts with "com.descartes.escheduler"
- those that are for the optimizer engine, RMPI

➲ **Note—** For the purposes of this guide, the RMPI properties should be set as usual and will not be discussed.

The following properties are the only “com.descartes.escheduler” properties that should be set. They define how scheduler communicates with Descartes Route Planner to get data (routes).

com.descartes.escheduler.update_from_lnosfw = 1

> This property enables the scheduler so that the BGO can receive its data from Descartes Route Planner.

com.descartes.escheduler.lnosfw.worker_type = BGO

> This property specifies whether the scheduler is dynamic or dedicated. The valid values are:
>
> - BGO (dynamic)
> - DEDICATED_BGO

com.descartes.escheduler.lnosfw.filter_key = DefaultBGO

> This property is only set for a Dedicated BGO. It is the value of the DataSetGroupKey of the data slice to use.

com.descartes.escheduler.lnosfw.ignore_area = 1

> This property is currently not used and should either be commented out or set to 1.

com.descartes.escheduler.lnosfw.log_changes = 1

> If set to 1, the xml of any improvement suggestions sent to Descartes Route Planner will be logged to a sub folder of the Fleetwise/scheduler/conf/area/logs folder. This property is only useful for troubleshooting, so set to zero in production.

com.descartes.escheduler.source_url = http://<machine name>/LNOS%20FW%20UI/Core/CtyXmlInterface/CtyInterfacesCCL.CtyXMLCC.asp

> This property specifies the URL of Descartes Route Planner to use for requesting data. In previous releases, if the BGO server was connected to a VPN while the BGO was running, the BGO was unable to connect to Descartes Route Planner to retrieve the data slice in some cases. To prevent this issue, the BGO has been enhanced to report the bad response received when attempting to retrieve the data slice. Additionally, the com.descartes.escheduler.source_url property can be set to use a URL with the server name as one of the following alternatives (replacing <machine name> above):
>
> - Localhost
> - 127.0.0.1
> - An IP address
>
> ➲ **Note—** Using localhost or 127.0.0.1 makes the local.properties files more portable between servers.

➲ **Note—** The installer currently only supports servername.

com.descartes.escheduler.copy_interval = 86400

This property should be left at this value or higher. It is not used for the moment.

com.descartes.escheduler.lnosfw.company_name = FW

com.descartes.escheduler.lnosfw.login_name = FW-Admin

com.descartes.escheduler.lnosfw.password = cs

These 3 properties specify the login information that scheduler should use when sending a request to Descartes Route Planner. The BGO instance can optimize more than one company (organization). This is accomplished by setting multiple

Com.descartes.escheduler.lnosfs.company_nameX =

Properties. The X on the end of name should be replaced with any character that makes this a unique property name. For example:-

com.descartes.escheduler.lnosfw.company_name1= SiouxFalls

com.descartes.escheduler.lnosfw.company_name2= Omaha

The BGO will alternate asking for a data slice for each company. The login name and password must be the same for all companies.

Optimizer.properties

When the BGO client service is started it will instruct scheduler to request a new set of data from Descartes Route Planner. The first property file that the BGO reads is the optimizer.properties from the Fleetwise/scheduler/conf/area folder. It can in turn point to other properties files; this feature is usually only used when scheduling the BGO to run different optimizations at different times of the day.

The optimizer.properties file is setup with default values by the installer which can be changed if the implementation requires it.

As of Descartes Route Planner version 15.2, most of the optimizer properties can be configured through the UI and linked to a data slice. Please see the section on BGO Properties in the database. In the following descriptions there is a comment next to those that cannot be configured through the UI and still have to be set in the optimizer.properties file.

Installation Properties

These properties are set by the installer and should not normally be changed. They cannot be configured through the UI.

com.descartes.escheduler.optimizer.live_url =

com.descartes.escheduler.optimizer.work_url =

They are both set to the url of the scheduler instance.

com.descartes.escheduler.optimizer.lnosfw_data = 1

This property sets the BGO to sync with Descartes Route Planner.

com.descartes.escheduler.optimizer.refresh_lnosfw_data = 3

This property defines how frequently the BGO will instruct scheduler to refresh the current set of routes held in memory from Descartes Route Planner. It is a number of minutes.

com.descartes.escheduler.optimizer.scheduler_service_name = FW_Scheduler_LNOS_areaxx

This is an optional property for the name of the scheduler service. The BGO client monitors the java scheduler and if for any reason there is a crash, it will restart the service if this property is set.

com.descartes.escheduler.optimizer.scheduler_service_temp_path =

This is an optional property for the path to the scheduler temp folder. If the BGO needs to restart the scheduler and this property is set, then it will delete all files in this folder except lme and lpe files before starting scheduler.

com.descartes.escheduler.optimizer.scheduler_restart = 120

This is an optional property. The BGO client can be set to restart java scheduler periodically. This is the number of minutes between restarts.

com.descartes.escheduler.optimizer.scheduler_service_map_edit_files_path =

This is an optional property. If set the map edit files will be copied to the area/temp folder whenever the scheduler service is restarted by the BGO.

Data Slice Optimization Time

com.descartes.escheduler.optimizer.new_lnosfw_data = 60

This property defines the number of minutes to optimize a data slice (set of routes) from Descartes Route Planner before requesting a new data slice. If this property is set to zero (Dedicated BGO), then the BGO will not ask for a new set of routes.

If the Data Slices select a varying number of routes then just using the new_lnosfw_data setting is not efficient because larger data slices require more optimization time than small data slices. So the following properties can be used in conjunction with new_lnosfw_data.

com.descartes.escheduler.optimizer.route_optimization_count =

com.descartes.escheduler.optimizer.assign_count =

These specify how many times each route should be optimized and how many times an attempt should be made to assign each unassigned order before moving on to the next data slice. If the time specified by new_lnosfw_data is reached before the routes have been optimized the requested number of times the BGO will move on to the next data slice.

Either or both of these properties may be set. If the route count is set but not the assign count then the BGO will run the assignment phase at least once before moving on to the next slice. In this case the BGO may or may not have attempted to assign all the unassigned once.

If both counters are set then the BGO keeps on running all the different kinds of optimization until the intra and Batch Assign/Inter optimizations have been run on each route at least the requested number of times and an attempt has been made to assign all the unassigned at least the requested number of times.

com.descartes.escheduler.optimizer.slice_complete_count =

This property determines how many times a BGO optimizes a slice without finding improvements before it notifies Descartes Route Planner that the slice has been optimized completely. Once flagged as completed Descartes Route Planner will not assign the slice to a BGO until either the routes or orders have been changed by another process or it is CtySysValues.BGOTimeout minutes since it was flagged as complete.

Changing Schedule Settings

The BGO can be set to use different Schedule settings than those set in the UI (in the FWSchedule table). For example, the BGO can run with a different inter chain size, number of measures or DynamicZoning settings than set in the UI.

com.descartes.escheduler.optimizer.set.xxxxxx =

Where xxxxxx is the RMPI name of an Optimizer setting. For example;

- com.descartes.escheduler.optimizer.set.RestrictToZonableJobs = 1
- com.descartes.escheduler.optimizer.set.OnlyAllowSwapsWithinZone = 0

If the DynamicZoningDistance is changed with this property, it will only affect the global schedule value not Dynamic Zoning Distance on the resources.

The BGO can be set to run with a smaller DynamicZoningDistance than is set on the resources and in the schedule settings. This setting can be useful to tightly cluster routes initially and then expand as last minute orders are received.

com.descartes.escheduler.optimizer.dynamic_zoning_distance_percent = 10

In this example, the schedule DynamicZoningDistance and the DynamicZoningDistance on each resource (if set) will be reduced by 10%. So 100 KM is reduced to 90 KM.

Below is a list of the Schedule Settings that can be overridden.

- DynamicZoning
- DynamicZoningDistance

- InterRouteChainSize
- IntraRouteChainSize
- Merges
- MergeThreshold
- PreRoute
- SwapsToNeighbors
- SwapsToNeighborsPoints
- SwapsWithinZone
- UnassignUnservicedOrders
- ViableOrders
- ZonableOrders
- NumberOfMeasures
- ImprovementThreshold

Scheduling Sessions

The BGO can be configured to run using different sets of properties at different times of the day. It can also be scheduled to sleep. These cannot be set through the UI.

```
com.descartes.escheduler.optimizer.schedule.xxxx.start
com.descartes.escheduler.optimizer.schedule.xxxx.end
com.descartes.escheduler.optimizer.schedule.xxxx.properties
```

These properties are set to enable different scheduler sessions. Multiple sets of these three properties can be used. The 'xxxx' in the property key can be any character string excluding spaces that is meaningful to the user and is used to tie the three properties together.

For example:

```
com.descartes.escheduler.optimizer.schedule.1.start = 09:00
com.descartes.escheduler.optimizer.schedule.1.end = 09:30
com.descartes.escheduler.optimizer.schedule.1.properties =
c:/SchedProps/noAssign.properties

com.descartes.escheduler.optimizer.schedule.zone3.start = 10:00
com.descartes.escheduler.optimizer.schedule.zone3.end = 14:00
com.descartes.escheduler.optimizer.schedule.zone3.properties =
c:/SchedProps/zone3.properties
```

The start and end times are military times with a ':' between the hours and minutes. The start and end times can pass midnight that is 13:00 to 08:00 is valid. As you can see from the examples the third property is the path to a file containing any of the optimizer.properties. i.e. properties starting with com.descartes.escheduler.optimizer. The properties in these files take precedence over properties in the main optimizer.properties file. So if normally the BGO is set to assign unassigned orders but for some reason between 09:00 and 09:30 it should not assign unassigned orders then the file noAssign.properties in the first example would contain just this property to turn assignment off.

    com.descartes.escheduler.optimizer.assign = 0

The third property can be set to the word sleep rather than the path to a properties file in which case the BGO will sleep between the start and end times.

    com.descartes.escheduler.optimizer.schedule.X.properties = sleep

The BGO will use the default properties when there is not a set of schedule properties for that time.

    com.descartes.escheduler.optimizer.default_sleep = sleep

This property can be used to set the default behavior of the BGO to sleep.

Selecting Resources

A BGO usually optimizes all the routes passed to it in a data slice but it can be configured using these properties to optimize a subset of those routes.

com.descartes.escheduler.optimizer.resource_attribute

com.descartes.escheduler.optimizer.resource_attribute_value

com.descartes.escheduler.optimizer.resource_attribute_comparison

These properties can be used to restrict BGO to optimize a subset of resources. The attribute comparison property is optional. If it is not set, then EQUALITY is assumed. If set, the following values can be used:

- EQUALITY
- NOT_EQUAL
- LESS_THAN
- GREATER_THAN
- GREATER_THAN_OR_EQUAL
- LESS_THAN_OR_EQUAL
- NOT_NULL

The attribute value can be one of the following three:

- An actual value enclosed in double quotes.
- The string today or today+n not enclosed in quotes, where n is an integer number of days. For example:

  com.descartes.escheduler.optimizer.resource_attribute = _deliveryDay
  com.descartes.escheduler.optimizer.resource_attribute_value = “123456 hjk”

com.descartes.escheduler.optimizer.resource_attribute_value = today
com.descartes.escheduler.optimizer.resource_attribute_value = today+3

The optimizer will convert today+n to today's date plus n days. That date will be used as a string in the form yyyymmdd to compare to the attribute of the resource specified. Only resources and their routes with that attribute set will be used in the optimization.

- The string date_today or date_today+x not enclosed in quotes. This is the same as number 2 except that the date is converted to mm/dd/yyyy. This can be compared with a resource date attribute that is a String or Date object inside Scheduler.

com.descartes.escheduler.optimizer.zone

This property can be used to further restrict the resources by setting a list of zones. The value for this property can be a list of zone names separated by a comma. For example:

com.descartes.escheduler.optimizer.zone = "NY",CT,NJ

The zone names can be enclosed in double quotes for readability and spaces in the list are significant. This cannot be set through the UI since it is not recommended.

Selecting Unassigned Orders

The BGO usually attempts to assign all the unassigned orders in a data slice but these properties can be used to select certain unassigned orders/jobs from the data slice.

com.descartes.escheduler.optimizer.job_attribute

com.descartes.escheduler.optimizer.job_attribute_value

com.descartes.escheduler.optimizer.job_attribute_value_comparison

These properties can be used to restrict BGO to optimizing a subset of the unassigned jobs.

The attribute comparison property is optional. If it is not set, then EQUALITY is assumed. If set, the following values can be used:

- EQUALITY
- NOT_EQUAL
- LESS_THAN
- GREATER_THAN
- GREATER_THAN_OR_EQUAL
- LESS_THAN_OR_EQUAL
- NOT_NULL

The attribute value is a string enclosed in double quotes.

com.descartes.escheduler.optimizer.assign_zone

This property allows unassigned jobs to be selected by zones by specifying a list of zones. The value for this property can be a list of zone names separated by a comma. For example:

com.descartes.escheduler.optimizer.zone = "NY",CT,NJ

The zone names can be enclosed in double quotes for readability and spaces in the list are significant.

Properties for Assigning Unassigned Orders

com.descartes.escheduler.optimizer.assign = 1

Setting this property > 0 turns on the assignment of unassigned orders.

com.descartes.escheduler.optimizer.assign_use_getSuggestion = 1

Setting this property to 1 makes the BGO use a getSuggestion optimization call to RMPI rather than a suggest (assignment) call.

com.descartes.escheduler.optimizer.assign_max_routes = 5

If set the BGO will select the closest N routes to the unassigned orders rather than all compatible ones. This will speed up the assignment optimizations.

When assigning orders, the assign_max_routes setting can be used to limit the number of routes considered for each assignment optimization.

com.descartes.escheduler.optimizer.assign_profit_order= 1

If set the BGO will assign the unassigned orders in descending profit order.

com.descartes.escheduler.optimizer.assign_run_intra = 1

When running an assignment turn Intra (resequence) optimization on as well. This will cause an Intra to be run on all the routes selected for consideration for each assignment of an order or group of orders.

Please note the following regarding this setting:

- If the set of routes is not restricted by the assign_max_routes, dynamic zoning, territory or other related factors, this change could increase optimization time for large numbers of routes.
- The Intra phase can be disabled while unassigned orders are being assigned.
- A call to assign an order could result in an improvement that implements the Intra improvement without actually assigning the order.

com.descartes.escheduler.optimizer.assign_location_order= 1

If set the BGO will group unassigned orders in geographical clusters and assign the most difficult clusters first.

The clusters are based on the same longitude to one decimal place and then the same latitude to 2 decimal places. These default values can be changed to adjust the geographical size of the cluster with these properties:-

com.descartes.escheduler.optimizer.assign_location_order_decimal_places_longitude = 2

com.descartes.escheduler.optimizer.assign_location_order_decimal_places_latitude = 2

Five new properties have been added to control the clustering of double-ended and multi-task orders. If these properties are not set, the handling of these orders will function as in previous versions as described above.

For both double-ended and multi-task orders:

com.descartes.escheduler.optimizer.assign_location_order_all_locations =

If this property is set, the other new properties described below are ignored. Default setting is 0. If set to "1", the BGO looks at all locations referenced by single, double-ended and multi-task orders and groups by the order location referenced the most. For example, if a double-ended order has its pickup location referenced by three other orders and its dropoff referenced by six other orders then the double-ended order will be placed in the group of six and removed from the group of three.

The following two properties are for double-ended orders:

com.descartes.escheduler.optimizer.assign_location_order_DE_prefer_customer =

1: The default in previous versions. If one of the locations is a customer, that location will be used

0: Does not check for a customer location

com.descartes.escheduler.optimizer.assign_location_order_DE_cluster_dropoff =

1: The default in previous versions. Cluster by dropoff location.

0: Cluster by pickup location

For example, to always cluster by the pickup location of a double-ended order, regardless of whether it is a customer or depot both properties would be set to "0".

assign_location_order_DE_prefer_customer = 0

assign_location_order_DE_cluster_dropoff = 0

The following two properties have been added for multi-task orders:

com.descartes.escheduler.optimizer.assign_location_order_multi_cluster_dropoff =

1: Cluster on dropoff location of multi-task

0: Do not cluster on dropoff location

com.descartes.escheduler.optimizer.assign_location_order_multi_cluster_rank = X

X: If cluster_dropoff not set to "1", then orders will be clustered on the task by X's value

➲ **Note—** To use the Rank property, the Rank must be sent in on the order when it is created.

The difficulty of a cluster can be the number of orders in the cluster or the total of the measures in the cluster. (dropoff +, pickup -). The default is the number of orders in a cluster and this property changes the difficulty determining factor to measures.

com.descartes.escheduler.optimizer.assign_location_order_capacity = 1

The measures of the orders in a cluster group are totaled (+ for dropoffs - for pickups). The MeasureN is multiplied by MeasureNFactor if factor is set. When determining the order of the clusters the total for each measure is compared starting with measure 1 then 2 then 3 up to 9.

When comparing measures if they are within X% of the resource capacity for that measure they are considered equal and the next measure is checked.

So if within Resource.MeasureN * (X/100). Default is 10% and it can be changed by property:-

com.descartes.escheduler.optimizer.assign_location_order_capacity_percentage = 5

If assign in descending profit order is also set then the clusters are for orders of the same profit and so the order of assigning is profit then cluster difficulty.

If assign in descending profit and location order are both set then setting CombineCustomers can be used. This makes the BGO combine orders at the same latitude and longitude with windows that overlap, at the highest profit of those orders. Double ended orders are not combined.

com.descartes.escheduler.optimizer.assign_location_order_combine_orders = 1

com.descartes.escheduler.optimizer.assign_location_order_min_group=

Whether the difficulty of a cluster is set to be the number of orders in the cluster or the total of the measures in the cluster this setting specifies the minimum number of orders to be in a cluster before the BGO attempts to assign it. So if orders are trickling into Route Planner this can ensure that the initial assignments are clusters.

com.descartes.escheduler.optimizer.assign_location_order_min_group =

This setting makes the BGO turn on the Inter and Intra optimizations settings for each assignment call when the BGO is attempting to assign a group of orders > = this number. This effectively runs an Optimize All on the group of unassigned orders and the routes selected as possible candidates for them. So this will make each optimization call longer but result in better routes.

com.descartes.escheduler.optimizer.assign_group =

If assign_location_order is not being used unassigned orders can still be assigned in groups rather than one at a time with this property. If it is set to greater than one, then that is the maximum number of jobs that will be assigned in one optimize call to Scheduler.

➲ **Note—** Only jobs with the same zone will be included together in a group unless Scheduler is set to ignore zones (Scheduler.IgnoreZones=1).

com.descartes.escheduler.optimizer.assign_group_minimum

This property changes the assign_group size to a minimum group size. When it is set to 1, the optimizer will wait until the number of jobs specified by assign_group are present before assigning them.

com.descartes.escheduler.optimizer.force_assign = 0

Normally orders are only assigned if the profit of the schedule increases, the assumption being that a resulting negative profit would generate a violation. The BGO can be forced to assign orders that generate a negative profit by setting this property. Also the Allowable Profit Drop in the CtySysValues table must set to the maximum negative profit drop allowed to limit the severity of violations that could be generated.

com.descartes.escheduler.optimizer.assign_exclude_gtr_dz = 1

The BGO can be configured to use a reduced Dynamic Zoning distance during optimization. This setting can be used to keep routes more compact for future days. Double-ended or multi-task orders can break this reduced Dynamic Zoning distance; the system is designed to assign these orders to a route of their own. The system then fills the rest of the routes and prevent other orders from being assigned.

This new setting in the BGO is designed to not assign a double-ended or multi-task order if the order exceeds the Dynamic Zoning distance. These orders will be left unassigned. This property is disabled by default.

Properties for Batch Assign / Inter

com.descartes.escheduler.optimizer.batch_assign = 1

com.descartes.escheduler.optimizer.inter = 1

These properties turn on Batch Assign and / or Inter if > 0. Normally they are both turned on.

com.descartes.escheduler.optimizer.inter_group = 3

com.descartes.escheduler.optimizer.inter_group_empty_routes = 2

com.descartes.escheduler.optimizer.inter_group_empty_routes_max = 5

The inter_group property defines the number of compatible used routes to be grouped together for a Batch Assign or Inter optimization. If the Inter_group_empty_routes property is 0 then no empty routes will be included in the group. If it is > 0 then the BGO will attempt to find this many empty routes to add to the group. If inter_group_empty_routes_max is set then the BGO can include up to this number of empty routes while searching for compatible used routes for the group. So with the settings above the group will include 3 used routes plus at least 2 empty routes but not more than 5 empty routes.

com.descartes.escheduler.optimizer.date_shift = 23

Routes are considered compatible for a group if their Earliest Start Dates are <= date_shift hours apart. So if orders have only one day time windows there is no point having routes for different days together in a group since no orders can be swapped between them.

com.descartes.escheduler.optimizer.batch_assign_neighbors = 1

Normally a group of compatible routes is formed for the target route by selecting randomly from the set of routes until the correct number have been found. If this property is turned on then the BGO will calculate the centroid of each route so that when forming groups of routes it will select the closest routes first for a group. When groups have been formed and optimized using the closest routes then the BGO will revert back to selecting routes randomly for groups. The usual criteria for determining if routes are compatible and so can be placed in the same group still applies. i.e. they must be for the same territory, date (based on the date_shift property) and have an order within the DZ distance of each other. Default is for this feature to be off.

com.descartes.escheduler.optimizer.batch_assign_use_all_unassigned = 0

com.descartes.escheduler.optimizer.batch_assign_use_all_unassigned_max= 10

The Batch Assign optimization normally uses just the orders that were already assigned to the routes in the group. If this property is set then all the unassigned orders will be included as well. This means that some orders that were assigned could become unassigned, they could be replaced by an unassigned order. If there could be a large number of unassigned orders then the use_all_unassigned_max property should be set to limit the number used otherwise the optimization could take a long time.

com.descartes.escheduler.optimizer.batch_assign_allow_unassign

This property can be used to ensure that no jobs are ever unassigned during the Batch Assign phase. It must be set to 0 to stop any orders being unassigned. A job might become unassigned if it is causing a violation such that the penalty associated with it is less than the schedule MinAssignmentProfit value, although this does not happen frequently. The default behavior is to allow unassignment of jobs with severe penalties. If batch_assign_use_all_unassigned is turned on then this property is automatically turned off.

Properties for Resequence / Intra

com.descartes.escheduler.optimizer.intra = 1

This property turns on the resequence optimizations of routes if > 0. The routes are resequenced one at a time and any improvements set to Descartes Route Planner immediately.

com.descartes.escheduler.optimizer.intra_batch_assign = 1

If this property is set to 0 or not set at all then the BGO just runs a RMPI intra optimization on the route. If this property is set to 1 the BGO will unassign all the stops on a route and batch assign them back on to that route again. This could unassign some orders if Schedule.UnassignUnserviced =1 or the resulting penalty generated by an order is < schedule MinAssignmentProfit.

Properties for Checking Multiple Routes Servicing Same Location.

com.descartes.escheduler.optimizer.check_route_loc = 1

This property turns on the Check Route Location optimizations if > 0. The BGO will find and attempt to fix all instances of more than one route servicing the same location on the same day.

Property for Checking Order Date Violations.

com.descartes.escheduler.optimizer.check_order_date_violation = 1

This property turns on the Check Order Date Violation optimization if > 0. The BGO will find and unassign all orders on a route for the wrong date.

Properties for Fill Routes

com.descartes.escheduler.optimizer.fill = 1

This property turns on the Fill Routes phase.

com.descartes.escheduler.optimizer.fill_start_distance = 10000

com.descartes.escheduler.optimizer.fill_end_distance = 20000

Start and end distance in meters, the default value is 10000 to 20000.

Using an example of 5000 to 35000, the process is as follows:

1. For each route the BGO will find all unassigned orders within 5000 meters of the existing stops on the route. It will run a suggest RMPI call on the unassigned order to make sure this route is the best one for the order.
2. The BGO will then increase the distance and run step 1 again for each route. The starting fill distance is used as the first increase and then doubled each time. So if the start fill distance is 1000 meters and the end fill distance is 32000 then the fill will be run for 1000, 2000, 4000, 8000,16000, 32000. If the start and end fill distances were 5000 to 4000 then it would be run for 5000, 10000, 20000, 40000

com.descartes.escheduler.optimizer.fill_depot_distance_start_percent = 50

com.descartes.escheduler.optimizer.fill_depot_distance_end_percent = 50

The BGO calculates the straight line distance of the existing stops on a route from the origin depot. It then takes the maximum and uses that value in conjunction with the fill_depot_distance properties. The filling process described in steps 1 and 2 above is run in an attempt to assign orders whose straight line distance from the origin depot is at least X percent of the maximum. So if the default is used of start and end equals 50 percent, then only those unassigned orders at least 50 percent of the maximum will be used for filling. If the start percentage is 40 and the end percentage is 0, then steps 1 and 2 will be run first at 40 percent then decreased by 20 points and run again at 20 percent. So with a start and end of 40 percent and 0 percen respectively, steps 1 and 2 will be run at 40 percent, 20 percent and 0 percent.

com.descartes.escheduler.optimizer.fill_repeat = 2

The default value is for the fill phase to run through all progressions of distance and percent only once for a data slice and then not again until the BGO cycles around to begin the optimization of the data slice again. This behavior can be overridden with the fill_repeat property, which specifies how many times the system should run the fill phase. This property is useful in cases where, if the assign phase is turned on to populate empty routes, the routes will then be included in the filling phase.

com.descartes.escheduler.optimizer.fill_earliest_days = 2

The fill_earliest_days property specifies the number of days of which used routes should be filled in the data slice. If the data slice contains routes for five days but only the first two days of routes should be filled, users would set this property to "2". The BGO inspects the routes to determine the earliest start date of all routes and then fills that many days.

For example, if there is an unassigned, low priority order for Customer XYZ that is close to a stop on Route 123 that is being filled, but there is already a high priority order for Customer XYZ assigned to a route in the future that is not being filled, then the order will not be assigned to the route 123. The RMPI suggest call will return the route in the future that is already going to that customer.

com.descartes.escheduler.optimizer.fill_intra = 5

Specify how many orders the system can assign to a route before running an intra (resequence). The default value is "5". The more frequently a route is resequenced while being filled will improve route quality but slow down the filling process. If this property is set to 0 then no resequences will be run. This is useful when optimizing a dispatching schedule.

com.descartes.escheduler.optimizer.fill_force = 0

When assigning an order to a route, the resulting profit may decrease because a new recharge has been generated. If fill_force property is set to "1", the BGO will still assign the order as long as it does not generate a violation and it is currently working on the start fill distance. For this to be accepted by Descartes Route Planner, the CtySysValues keyword AllowableProfitDrop must be set to a large number (usually 1000+ depending on costs). This property is disabled by default.

com.descartes.escheduler.optimizer.fill_exit = 1

Enable this property to have the BGO cycle round to the next data slice when the filling is complete. The default is 0, i.e. do not exit.

Properties for Assign Matching

com.descartes.escheduler.optimizer.assign_matching = 1

Default is for Assign Matching to be off (0)

Properties for Optimize All

com.descartes.escheduler.optimizer.optimize_all = 1

When this property is set to 1 the BGO will run an Optimize All internally on the routes and orders selected by the data slice. The database is not locked as it would be by running this from the UI.

com.descartes.escheduler.optimizer.optimize_all_unassign = 1

If this property is set then the BGO will unassign all orders internally before running the Optimize All. The default behavior is not to unassign the orders.

com.descartes.escheduler.optimizer.optimize_all_exit = 1

If this property is set then the BGO will request the next data slice when the optimization is finished. If it is not set (default behavior) then the BGO will start running incremental optimizations according to the other properties in optimizer.properties until it is time for a new data slice.

com.descartes.escheduler.optimizer.optimize_all_inter = 0

The optimize_all_inter property can be used to disable all Inter optimizations in cases where users need an Optimize All-type optimization o complete more quickly. Normally the RMPI optimizer is set to use Assign, Intra and Inter during Optimize All, but, comparatively, the Inter optimizations take the longest amount of time. Inter optimizations can be turned off by setting this property to "0". By default, this property is enabled.

➲ **Note—** Route quality will be reduced without Inter optimizations.

Phase Time Properties

These properties control how long each kind of optimization phase is run before moving on to the next kind of optimization. The values are the number of minutes to run a particular kind of optimization phase.

com.descartes.escheduler.optimizer.assign_phase_time = 3

Length of time to spend assigning unassigned orders before cycling round.

com.descartes.escheduler.optimizer.inter_phase_time = 10

Length of time to spend on Batch Assign / Inter optimizations before cycling round.

com.descartes.escheduler.optimizer.intra_phase_time = 5

Length of time to spend running resequence optimizations before cycling round.

com.descartes.escheduler.optimizer.check_route_loc_phase_time = 5

Length of time to spend checking multiple routes servicing the same location before cycling round.

com.descartes.escheduler.optimizer.fill_phase_time = 15

Length of time to spend running Fill route optimizations before the BGO cycles round. Whatever the value of the fill phase time, the BGO will not exit the fill phase to cycle round other optimizations (Batch Assign, Assign etc) until it has tried to fill every route using the starting fill distance.

com.descartes.escheduler.optimizer.assign_matching_phase_time =   (minutes)

If the phase time is not set then the BGO will attempt to match all the unassigned before moving on.

Optimization Timeout Properties

When running incremental optimizations it is important that each optimization does not take too long so that the risk of any improvement being rejected is reduced. These properties set a timeout (in seconds) for each kind of optimization. If the optimization call to RMPI exceeds this then the optimization is cancelled and the BGO moves on to the next optimization.

com.descartes.escheduler.optimizer.assign_run_time = 300

Timeout for each assign optimization.

com.descartes.escheduler.optimizer.batch_assign_time = 300

Timeout for each Batch Assign optimization on a group of routes.

com.descartes.escheduler.optimizer.inter_run_time = 300

Timeout for each Inter optimization on a group of routes.

com.descartes.escheduler.optimizer.intra_run_time = 400

Timeout for each resequence optimization on a route.

Order of Incremental Optimizations

Normally the BGO runs the optimization phases in this order:-

Intra

Batch Assign / Inter

Assign unassigned orders

Check multiple routes servicing the same location.

Check Order Date Violations

Fill

Assign Matching

It runs each phase for the time set in the phase time properties and then if there is still more time left to optimize the current data slice, it starts running the phases again. The order in which the phases are run can be changed by setting the properties that turn optimization phases on or off to values greater than 1. So for example having the properties set like this:-

com.descartes.escheduler.optimizer.intra = 4

com.descartes.escheduler.optimizer.inter = 2

com.descartes.escheduler.optimizer.assign = 1

com.descartes.escheduler.optimizer.check_route_loc = 3

would run the phases in this order:-

Assign unassigned orders

Batch Assign / Inter

Check multiple routes servicing the same location

Intra

Properties to Specify Which Settings to use with a Data Slice

A BGO instance can be set to use different properties depending on the Data Slice name. So some Data Slices might need the Batch Assign optimizations run and others not need orders swapped between routes. These properties cannot be set through the UI. Their functionality has been superseded by the existing of BGO properties sets in the database. See the section on BGO properties in the database.

com.descartes.escheduler.optimizer.data_slice_name.X =

com.descartes.escheduler.optimizer.data_slice_name_path.X =

These two properties define which data slices can be optimized by which properties. The X on the end of the name should be replaced by a text string to make the property key unique.

If the value of the data_slice_name property is "contained in " the data slice name to be optimized then the properties defined by the path in property data_slice_name_path are used to optimize it.

Multiple data slice name properties can be used and a unique identifier on the end of the property name is used to link it to the data slice name path property. For example:-

com.descartes.escheduler.optimizer.data_slice_name.1 = branch10 com.descartes.escheduler.optimizer.data_slice_name_path.1 = c:/props/branch10.properties

com.descartes.escheduler.optimizer.data_slice_name.2 = branch13

com.descartes.escheduler.optimizer.data_slice_name_path.2 = c:/props/branch13.properties

So Data Slice names that contain "branch10" will have the properties in file branch10.properties added to the default values in optimizer.properties and used for optimizations. Data Slice names containing "branch13" will use branch13.properties.

The path to specific properties for the data slice does not have to be set. It is only necessary if there are properties different from the defaults.

Specifying Data Slices Not to Optimize.

com.descartes.escheduler.optimizer.not_data_slice_name.X=

This property defines a data slice that should not be optimized by this BGO instance. This property is most useful when used in conjunction with the scheduling properties so a Data Slice is not optimized at a certain time of day.

Multiple not_data_slice_name properties can be defined with a unique identifier on the end replacing the X. For example:-

com.descartes.escheduler.optimizer.not_data_slice_name = branch1

com.descartes.escheduler.optimizer.not_data_slice_name.1 = branch2

com.descartes.escheduler.optimizer..not_data_slice_name.2 = branch3

Properties to Enable Compact Routes Phase

The Compact Routes phase will attempt to reduce the number of routes being used. The Compact Routes phase is enabled with the following property. This phase is disabled by default.

com.descartes.escheduler.optimizer.compact_routes = 1

The BGO is designed to search for "short" routes as determined by two properties.

com.descartes.escheduler.optimizer.compact_routes_min_stops = 10

com.descartes.escheduler.optimizer.compact_routes_elapsed_percent = 15

A route is considered "short" if the route has less stops than the min_stops value or there is more spare elapsed time than elapsed_percent of the total available elapsed time of the route. If the difference between Resource.LatestEnd and Resource.EarliestStart is 10 hours and elapsed_percent is 10, then the route will be considered short if it is less than nine hours long.

For each short route, the BGO is designed to find other routes from the same depot and select a number of them as determined by the max_group property. It selects these routes based on whether or not there is a stop at the same latitude and longitude as the short route and then, if necessary, adds the shortest from the same depot until max_group routes are selected.

```
com.descartes.escheduler.optimizer.compact_routes_max_group = 5
```

The BGO then unassigns the orders from the short route and runs a mini-optimize all on the now empty short route, the selected group of routes and the unassigned orders from the short route.

The BGO may need two attempts to free up a route with this phase. The first time through, the route is shortened and the second time emptied altogether.

The following properties determine how long the phase is run before cycling around other phases. The run_time property determines how long an individual optimization is allowed to take.

```
com.descartes.escheduler.optimizer.compact_routes_phase_time = 5

com.descartes.escheduler.optimizer.compact_routes_run_time = 300
```

If defined, maxelapsedtime will be used instead of calculating LatestEnd to EarliestStart to retrieve the maximum elapsed time allowed.Configuring BGO Properties in the Database

In Descartes Route Planner version 15.2, the BGO properties contained in the optimizer.properties file that control the optimizations the BGO instance performs can be setup in the database via the Descartes Route Planner user interface. Multiple sets of optimizer properties can be generated that perform different optimizations and data slices can now be linked to those sets. This functionality allows more flexibility in how a data slice is optimized.

For example, users could set up the following case through the user interface:

- Data slice needs to be optimized with the usual BGO incremental optimizations from 00:00 to 17:00
- From 17:00 to 18:00, only Resequence and Assignment need to be run
- From 18:00 to 00:00, no optimization at all

Data Slice Sets

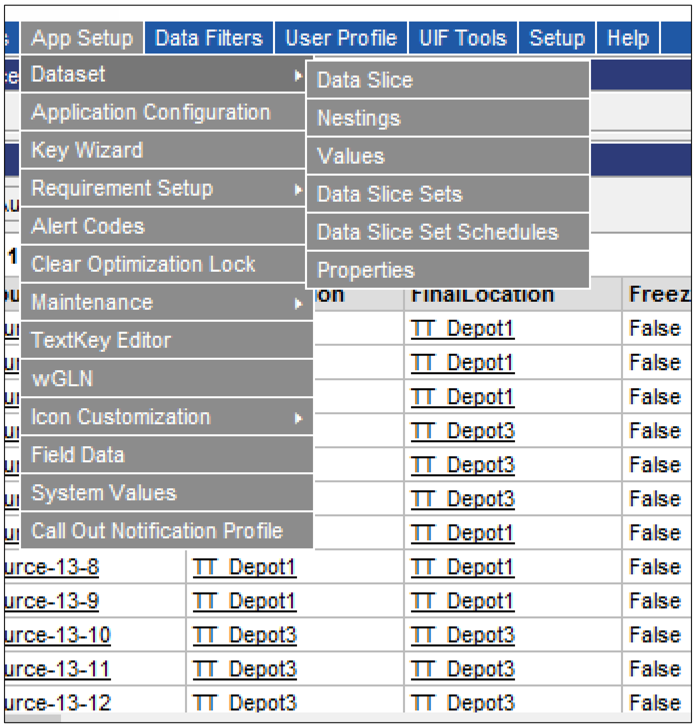
Data Slice Sets allow customers to group similar Data Slices together into a set and then specify how the set should be optimized. Customers usually have many data slices that need to be optimized in the same way. A customer using reservations will probably have data slices selecting routes for "today+1", "today+2" and the future for each branch. In some cases, the "today+1" routes might need to be optimized differently than the others. Users could allocate these routes to one Data Slice Set and the rest to another set. The "today+1" Data Slice Set could be setup to stop optimizing the data slices at 18:00 when they are moved out of planning and set to start again at midnight.

To summarize, a Data Slice Set is configured to use the appropriate BGO properties at certain times of the day and Data Slices are assigned to a Data Slice Set.

If a data slice is linked to Data Slice Set that is scheduled to use a set of optimizer properties from the database, then the database properties are used by the BGO instead of the ones in the actual Optimizer.properties file. If a data slice is not linked to a Data Slice Set, the BGO will use the actual optimizer.properties file when optimizing this slice. This process allows for backward compatibility so that when upgrading to Descartes Route Planner 15.2, the BGO properties and Data Slice Sets do not have to be setup in the database immediately and the BGO will continue to work as before. Additionally, the BGO can handle situations where some data slices are linked to Data Slice Sets and some not by switching back and forth between them. This functionality allows the change to using the optimizer properties in the database to be a gradual process.

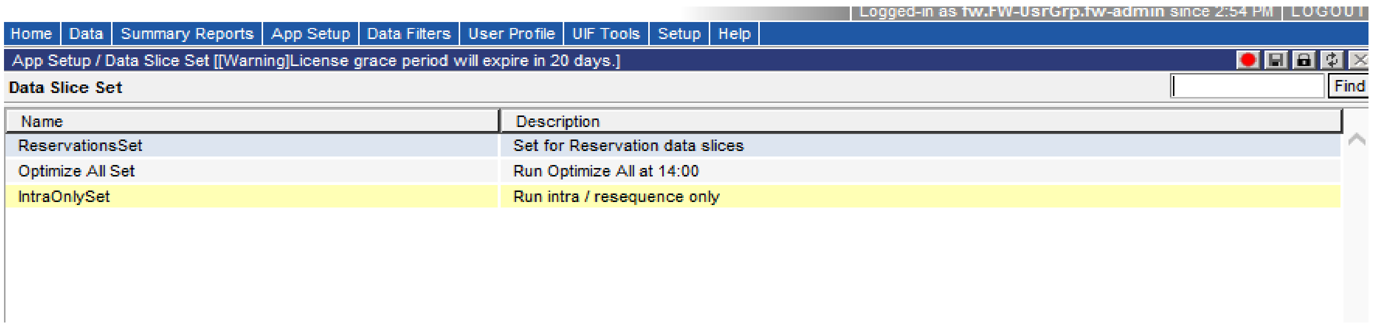
Menus

The **App Setup > Dataset** menu includes the Data Slice Sets and Properties options. The data slices are defined in the same way, using Nestings and Values.



	ion	FinalLocation	Freez
		TT_Depot1	False
		TT_Depot1	False
		TT_Depot1	False
		TT_Depot3	False
		TT_Depot3	False
		TT_Depot3	False
		TT_Depot1	False
urce-13-8	TT_Depot1	TT_Depot1	False
urce-13-9	TT_Depot1	TT_Depot1	False
urce-13-10	TT_Depot3	TT_Depot3	False
urce-13-11	TT_Depot3	TT_Depot3	False
urce-13-12	TT_Depot3	TT_Depot3	False

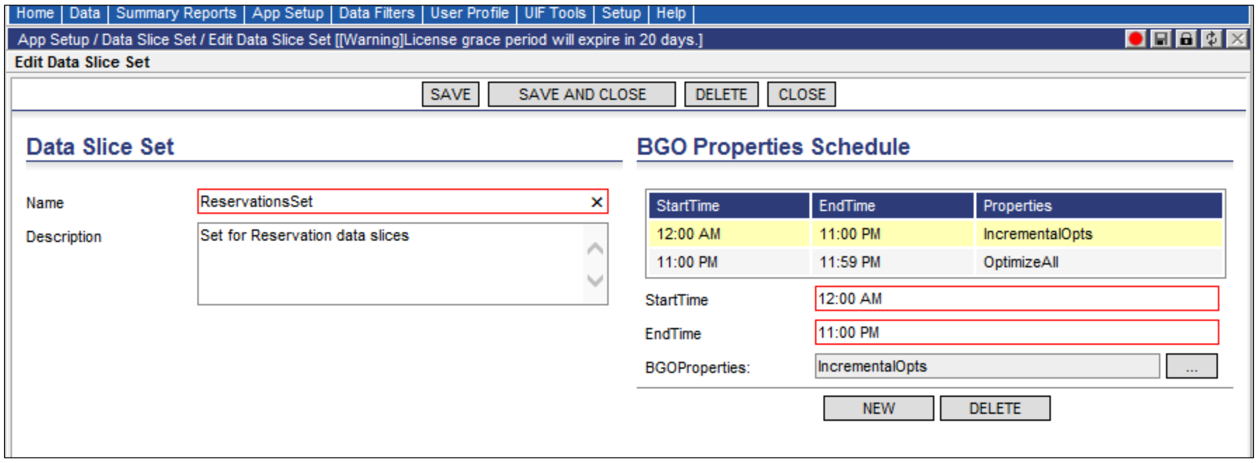
In the following image, three Data Slice Sets have been setup.


Logged-in as fw.FW-UsrGrp.fw-admin since 2:54 PM | LOGOUT
Home | Data | Summary Reports | App Setup | Data Filters | User Profile | UIF Tools | Setup | Help
App Setup / Data Slice Set [[Warning]License grace period will expire in 20 days.]
Find

Data Slice Set

Name	Description
ReservationsSet	Set for Reservation data slices
Optimize All Set	Run Optimize All at 14:00
IntraOnlySet	Run intra / resequence only

The ReservationsSet has been configured to use the BGO properties called IncrementalOpts from midnight to 11:00 pm and then run Optimize All on the data slices for an hour.


Home | Data | Summary Reports | App Setup | Data Filters | User Profile | UIF Tools | Setup | Help
App Setup / Data Slice Set / Edit Data Slice Set [[Warning]License grace period will expire in 20 days.]
SAVE
SAVE AND CLOSE
DELETE
CLOSE
NEW
DELETE

Edit Data Slice Set

Data Slice Set

Name: ReservationsSet

Description: Set for Reservation data slices

BGO Properties Schedule

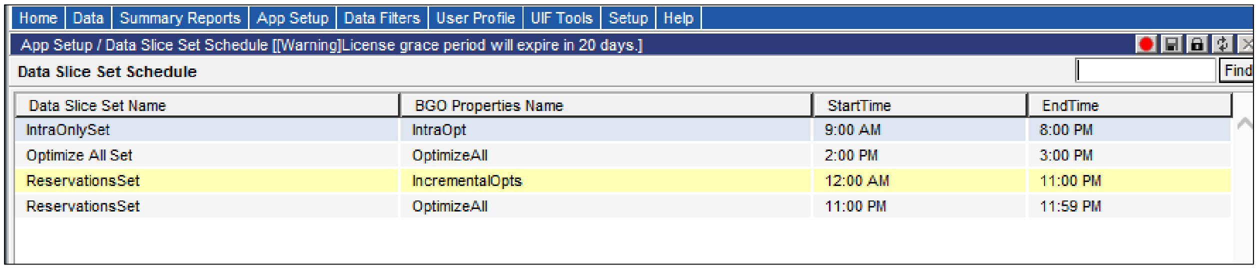
StartTime	EndTime	Properties
12:00 AM	11:00 PM	IncrementalOpts
11:00 PM	11:59 PM	OptimizeAll

StartTime: 12:00 AM

EndTime: 11:00 PM

BGOProperties: IncrementalOpts

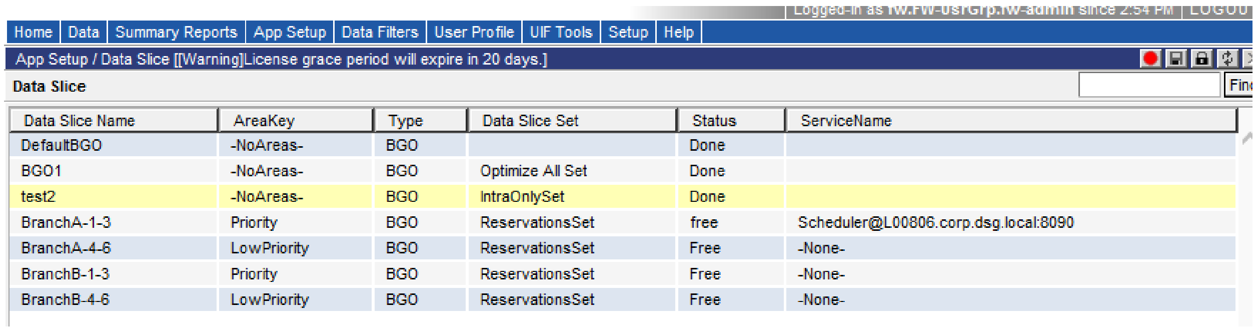
Rather than editing each Data Slice Set to view the configuration, users can select **App Setup > Dataset > Data Slice Set Schedules** to display a list of Data Slice Sets and associated properties. These items can be edited via right-click menu option.


Home | Data | Summary Reports | App Setup | Data Filters | User Profile | UIF Tools | Setup | Help
App Setup / Data Slice Set Schedule [[Warning]License grace period will expire in 20 days.]
Find

Data Slice Set Schedule

Data Slice Set Name	BGO Properties Name	StartTime	EndTime
IntraOnlySet	IntraOpt	9:00 AM	8:00 PM
Optimize All Set	OptimizeAll	2:00 PM	3:00 PM
ReservationsSet	IncrementalOpts	12:00 AM	11:00 PM
ReservationsSet	OptimizeAll	11:00 PM	11:59 PM

The Data Slices have been assigned to Data Slice Sets. In the image below, the data slice "DefaultBGO" has not been assigned to a Data Slice Set, so the actual optimizer.properties file of the BGO instance that optimizes the data slice will be used to determine which optimization to run.


Logged-in as rw.FW-UsrGrp.rw-admin since 2:54 PM | LOGOU
Home | Data | Summary Reports | App Setup | Data Filters | User Profile | UIF Tools | Setup | Help
App Setup / Data Slice [[Warning]License grace period will expire in 20 days.]
Data Slice
Find

Data Slice Name	AreaKey	Type	Data Slice Set	Status	ServiceName
DefaultBGO	-NoAreas-	BGO		Done	
BGO1	-NoAreas-	BGO	Optimize All Set	Done	
test2	-NoAreas-	BGO	IntraOnlySet	Done	
BranchA-1-3	Priority	BGO	ReservationsSet	free	Scheduler@L00806.corp.dsg.local:8090
BranchA-4-6	LowPriority	BGO	ReservationsSet	Free	-None-
BranchB-1-3	Priority	BGO	ReservationsSet	Free	-None-
BranchB-4-6	LowPriority	BGO	ReservationsSet	Free	-None-

Note that the AreaKey of a data slice is still used in the same way to allocate a number of BGO instances to data slices. Descartes Route Planner cycles round data slices as in previous versions, but if the data slice is assigned to a Data Slice Set and no properties are active at that time, then the Data Slice Set will not be allocated to a BGO instance for optimization. Using the examples above, any data slices assigned to the IntraOnlySet will only be passed to a BGO instance between 9 am and 8 pm. Outside of those times they will not be optimized.

Operating the BGO

Reading and Understanding Log Files

The BGO outputs a set of logs of its operations. The logs are found in each area folder. There are two log files: optimizer.log and scheduler.log for each area folder.

The following is an example extract from the optimization log file (optimizer.log).

```
22/05/2007 00:22   ",933 INFO  [OptimizerDriver   ] Intra-route on AD-C075-12708-20070523-BGD"  0:03:42

22/05/2007 00:26   ",026 INFO  [OptimizerDriver   ] Inter-route on [

                                                   AD-C075-12708-20070523-BGD,

                                                   AM-C075-12547-20070523-BGD,

                                                   AM-C035-12500-20070523-BGD

                                                   ] for zones [BGD]"   0:04:14

22/05/2007 00:28   ",227 INFO  [OptimizerDriver   ] Batch assign on [AD-C075-12708-20070523-BGD, AM-C075-12547-
20070523-BGD, AM-C035-12500-20070523-BGD] for zones [BGD]"      0:02:00
```

Each phase of the optimization will have a line informing you of the route or routes being optimized along with the time it started optimization.

The scheduler.log holds details of the loading of the Data Slice and the submission of improvements.

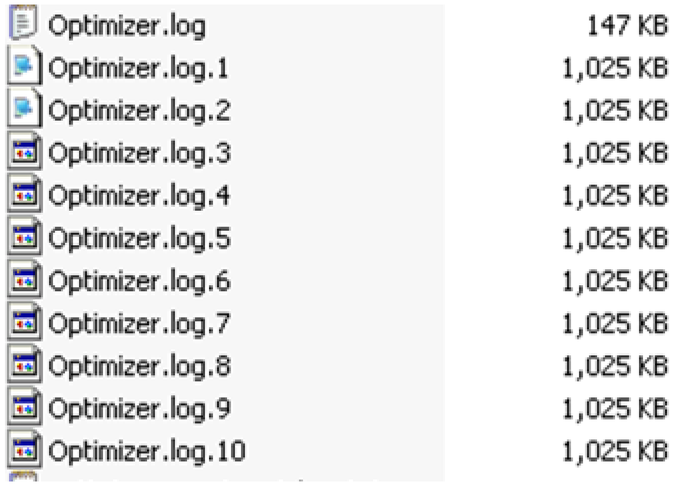
The following is an example extract from the scheduler log file (scheduler.log). In this example, an improvement was rejected because the routes had been changed by another process (UI).

```
2007-05-22 17:04:21,139 INFO  [CFScheduleAdapter R4814] Save Route:AD-C035-10084-20070524-BGH completed.

2007-05-22 17:04:21,514 INFO  [teSynchronization R4814] Error returned from LNOSFW ConfirmSchedule.

                              Resynching. Error message=FAILED –

                              Invalid Schedule Request:

                              DB Schedule Routes have been modified since last evaluation.

2007-05-22 17:04:21,514 INFO  [CFScheduleAdapter R4814] Requesting
LNOSFWServerURL=http://10.3.112.45/LNOS%20FW%20UI/Core/CtyXmlInterface/CtyInterfacesCCL.CtyXMLCC.asp

2007-05-22 17:04:28,108 INFO  [CFScheduleAdapter R4814] LNOS version is before 2.7.1.7

2007-05-22 17:05:09,670 INFO  [CFScheduleAdapter R4814] LNOSFW response:

                                              Number of Snapshot Elements Returned: 1832

2007-05-22 17:05:09,670 INFO  [CFScheduleAdapter R4814] LNOSFW response contains:

                                              Resources=478 Jobs=599 Customers=642 Depots=102
                                              Routing Params=9
```

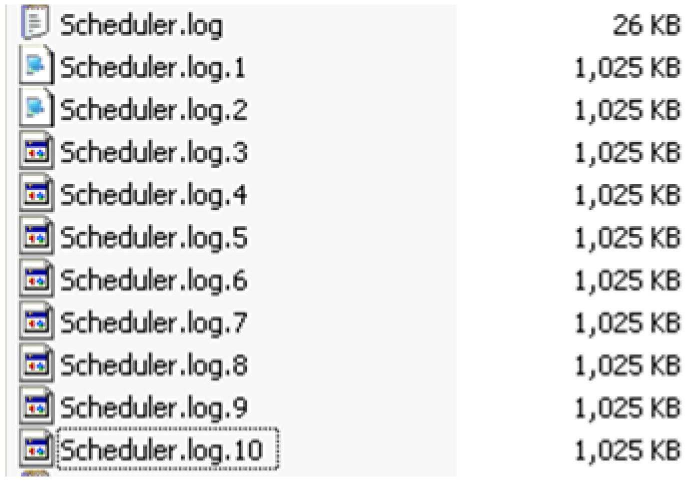
```
2007-05-22 17:05:09,670 INFO  [CFScheduleAdapter R4814] Sending edits to RMPI
2007-05-22 17:05:17,389 INFO  [EScheduler        R4814] Dropped elements
```

Properties to Control Log File Size.

The BGO writes two log files, optimizer.log and scheduler.log. The default configuration is to archive them when they reach 1mb and save up to 10 old files as shown below.



```
Optimizer.log        147 KB
Optimizer.log.1    1,025 KB
Optimizer.log.2    1,025 KB
Optimizer.log.3    1,025 KB
Optimizer.log.4    1,025 KB
Optimizer.log.5    1,025 KB
Optimizer.log.6    1,025 KB
Optimizer.log.7    1,025 KB
Optimizer.log.8    1,025 KB
Optimizer.log.9    1,025 KB
Optimizer.log.10   1,025 KB
```



```
Scheduler.log         26 KB
Scheduler.log.1    1,025 KB
Scheduler.log.2    1,025 KB
Scheduler.log.3    1,025 KB
Scheduler.log.4    1,025 KB
Scheduler.log.5    1,025 KB
Scheduler.log.6    1,025 KB
Scheduler.log.7    1,025 KB
Scheduler.log.8    1,025 KB
Scheduler.log.9    1,025 KB
Scheduler.log.10   1,025 KB
```

Once there are 10 archive files the oldest one will be deleted when another archive file needs to be created.

The number and size of archived log files can be changed using properties in local.properties and optimizer.properties.

For the scheduler.log files that are written by the scheduler service these properties in local.properties can be set.

com.descartes.escheduler.max_file_size =

com.descartes.escheduler.max_backup_index =

For the optimizer.log files written by the optimizer service these properties can be set in the optimizer.properties file; they cannot be set in the user interface.

com.descartes.escheduler.optimizer.max_file_size =

com.descartes.escheduler.optimizer.max_backup_index =

The value for max_file_size is the number of megabytes and must be an integer > 0 and <= 100, default is 1.

The value for max_backup_index is the number of archive files to keep, default is 10.

Reading and Understanding XML Sent and Received Files

The BGO's local.properties file has a property to turn on the logging of all the xml requests and responses between the BGO and LNOS. The property is:

com.descartes.escheduler.lnosfw.lob_changes = 1

➲ **Note—** When turned on, this property generates a lot of files. It is advised that it only be turned on for troubleshooting purposes.

As of Route Planner release 16.10 the xml logging can be turned on for specific Data Slices through the UI. When logging is on the Scheduler service will create a folder with the current date and time as its name in the area/logs folder or in the folder specified by this property if set.

com.descartes.escheduler.lnosfw.log_changes_path = d:/Descartes/BGOLogs/area1

➲ **Note—** The path must have forward slashes (/).

When Scheduler requests a copy of the schedule to optimize, the request xml will have a file name like 'GetScheduleRequest-1.xml'. The matching response will have a file name like 'GetScheduleResponse-1.xml'.

When the BGO finds an improvement the Scheduler service sends a request to Descartes Route Planner and receives either a success or failure. If the improvement request succeeded, then the request and response are written as 'SetSchedule-1-Request.xml' and 'SetSchedule-1-Response.xml'.

If the improvement was rejected for any reason, then the names of the files would be 'SetSchedule-1-Failed-Request.xml' and 'SetSchedule-1-Failed-Response.xml'.

Sending Command XMLs and Business Documents.

You can set the BGO to post Command xml or Business Document requests to Descartes Route Planner. The requests can be setup to perform any action that can be done from Descartes Route Planner. They have no connection to the data slice that the BGO might currently be setup to optimize. These requests cannot be setup in the user interface.

The requests are typically used to generate resources and buckets, move routes from one schedule to another, or perform an Optimize All in Descartes Route Planner rather than in the BGO. When used to perform an optimize all, the optimization is actually running in LNOS as if it had been initiated from the Descartes Route Planner UI with all the consequences of locking the database.

The URL to post to is defined by this property:-

com.descartes.escheduler.optimizer.commandXML_URL =

It should be set to this for command xml:-

com.descartes.escheduler.optimizer.commandXML_URL=http://(machine name)/LNOS%20FW%20UI/Core/CtyXmlInterface/CtyInterfacesCCL.CtyXMLCC.asp

➲ **Note—** Replace "machine name" with the machine where LNOS is located. (can be localhost)

For Business Documents it can be set to any Listener. For example:-

com.descartes.escheduler.optimizer.commandXML_URL=http://(machine name)/STAD/Listener/DocFWBOLListener.asp

The files to be posted to this URL are set with property:-

com.descartes.escheduler.optimizer.commandXML_X=

The X at the end of the property name is replaced with a unique number so that you can have more than one file posted to Descartes Route Planner. Below are two examples of paths to command xml files to be sent. They are numbered as in this example, and the BGO sends them in that order. The numbers do not have to be consecutive.

com.descartes.escheduler.optimizer.commandXML_01=C:/XML/MoveSched.xml

com.descartes.escheduler.optimizer.commandXML_2=C:/XML/GenResources.xml

If the response from posting a file to Descartes Route Planner is not Success then the BGO will not send the rest of the files. This behavior can be changed by setting this property to 1.

com.descartes.escheduler.optimizer.commandXML_continue_if_error = 1

Scheduling Posting Requests.

Normally these properties are used in conjunction with the Scheduling properties of the BGO. A user only needs the requests sent at a certain time of day; for example only generate new resources once a day between 04:00 and 04:10.

```
com.descartes.escheduler.optimizer.schedule.gen.start = 04:00

com.descartes.escheduler.optimizer.schedule.gen.end = 04:10
com.descartes.escheduler.optimizer.schedule.gen.properties =
c:/xml/GenResources.properties
```

The GenResources.properties file would contain something like this:-

```
com.descartes.escheduler.optimizer.commandXML_URL =
http://localhost/LNOS%20FW%20UI/Core/CtyXmlInterface/CtyInterfacesCCL.CtyXMLCC.asp

com.descartes.escheduler.optimizer.commandXML_1 =
C:/xml/generateResources_branch_93.xml

com.descartes.escheduler.optimizer.commandXML_2 =
C:/xml/generateResources_branch_95.xml

com.descartes.escheduler.optimizer.commandXML_4 =
C:/xml/generateResources_branch_99.xml

com.descartes.escheduler.optimizer.commandXML_sleep = sleep
```

The sleep property at the end of the list tells the BGO to sleep until the end time of 04:10 if it finishes sending the files before then. If not set to sleep the BGO will start running incremental optimizations on its data slice as soon as the files have been posted and not wait till 04:10.

Posting/Exporting Optimization Statistics and Route Information

A BGO instance optimizes a data slice for a certain length of time and then moves on to another one. There are settings that can ensure that it has attempted an optimization of every route and has attempted to assign every unassigned order before it moves on to the next data slice. Which optimizations it has attempted are controlled by the optimizer.properties.

The BGO has the following information that it can export about the data slice it has just finished optimizing.

1. The resource key of every route in the data slice with the order keys that are assigned to it.

2. The order key of every order left unassigned at the end of the optimization.
3. The order key of every unassigned order assigned during this optimization along with the resource key it was assigned to.
4. The resource keys of routes where an improvement was found in this BGO cycle.
5. The resource keys of routes that were optimized whether an improvement was found or not.
6. The statistics that are printed in the optimizer.log. Screen shot at the end of this document.
7. The Data Set Group Key
8. The current date and time.
9. The Company Name (org).

The BGO can be configured to export some or all of this information and it can either post it to a URL and/or write a file to a folder.

Implementation.

There are 4 properties for the optimizer.properties file to control the export.

1. com.descartes.escheduler.optimizer.post_statistics = 1
   turns this feature on and off. Default is off

2. com.descartes.escheduler.optimizer.post_statistics_template = c:/lnos/fleetwise/scheduler/customer-template.xml
   path to the customer's template file for the xml to post which can contain any or all of the %% strings defined later. This template is used to generate the xml which is then posted and/or written to a file.

3. com.descartes.escheduler.optimizer.post_statistics_url = http://<whatever>
   URL to post the export to.

4. com.descartes.escheduler.optimizer.post_statistics_folder = c:\<whatever>
   path for exporting the statistics to a file. The file name will have this format:-
   <org><data set group key><current date & time>.xml

Either or both the url and statistics folder properties can be defined as required but it is the responsibility of the customer to conserve disk space and delete the .xml files when they have been used. The success or failure of writing the file or posting the xml file will be in the optimizer.log.

XML Template File.

The xml template file can contain any or all of the following strings which will be replaced with the export data. A string can be defined more than once in the template and every occurrence will be replaced.

1. %Resources% replaced with:-
   ```
    <Resource ResourceKey=”abc” >
        <Order Orderkey=”123”  LocationKey=”cust555” />
         <Order Orderkey=”456” LocationKey=”cust999” />
   </Resource>
   <Resource ResourceKey=”qwerty” />   ← this would be an empty route →
   ```

2. %UnassignedOrders% replaced with :-
   ```
   <UnassignedOrder OrderKey=”3456” />
   <UnassignedOrder OrderKey=”99456” />
   ```

3. %OrdersAssigned% replaced with:-
   ```
   <OrderAssigned OrderKey="3456" ResourceKey="ABC" />
   <OrderAssigned OrderKey="99456" ResourceKey="Res-234" />
   ```

4. %ImprovedResources% replaced with :-
   ```
   <ImprovedResource ResourceKey=”abc” />
   <ImprovedResource ResourceKey=”xyz” />
   ```

5. %OptimizedResources% replaced with :-
   ```
   <OptimizedResource ResourceKey=”def”/>
   <OptimizedResource ResourceKey=”xyz”/>
   ```

6. %Statistics% replaced with:-
   ```
   <Statistics
   Type=”INTRA”   AvgTime=”792”   Count=”33”   ImprovementFound=”24”   Improvem
   entAccepted=”24” />
   <Statistics Type=”ASSIGN”  AvgTime=”1815” Count=”297” ImprovementFound=”292”
   ImprovementAccepted=”292” />
   <Statistics Type=”BatchAssign” ……………….etc   />
   <Statistics Type=”INTER” ……………….etc   />
   ```

7. %DataSetGroupKey% is replaced with just the data set group key. Nothing else, no <> or quotes.

8. %CurrentTime% is replaced with just the date time in this format yyyy-mm-dd hh:mm:ss. Nothing else, no <> or quotes.

9. %CompanyName% s replaced with just the Company Name (Org). Nothing else, no <> or quotes

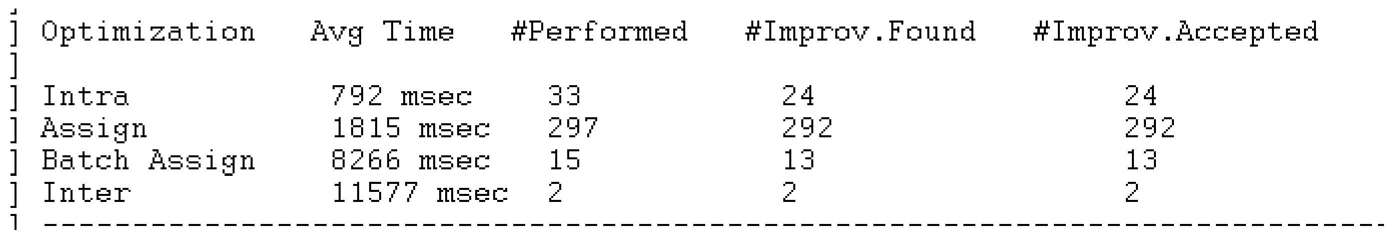
Sample template file

```
<?xml version = "1.0" ?>
<Command>
    <Header>
        <Authentication>
            <CompanyName>%CompanyName%</CompanyName>
            <GroupName></GroupName>
            <LoginName>fw-admin</LoginName>
            <Password>cs</Password>
          </Authentication>
          <Options/>
          <EchoData/>
          <SystemData/>
    </Header>
    <Execute>
          <Function>
                    <ApplicationCode>LNOSFW</ApplicationCode>
                    <MenuName>LNOS Reservations</MenuName>
                    <ModuleName></ModuleName>
                    <FunctionName></FunctionName>
         </Function>
```

```
            <Parms>
              <Input>
                  <DataSlice DataSetGroupKey="%DataSetGroupKey%"
       CurrentTime="%CurrentTime%" />
                  <Data>
                       %Resources%
                       %UnassignedOrders%
                  </Data>
                  <Optimization>
                       %ImprovedResources%
                       %OptimizedResources%
                       %Statistics%
                  </Optimization>


              </Input>
            </Parms>
          </Execute>
       </Command>
```

Statistics.

Screenshot of the statistics that are printed in the optimizer.log



```
] Optimization    Avg Time    #Performed    #Improv.Found    #Improv.Accepted
]
] Intra             792 msec     33            24                  24
] Assign            1815 msec    297           292                 292
] Batch Assign      8266 msec    15            13                  13
] Inter             11577 msec   2             2                   2
] ---------------------------------------------------------------------------
```